



PATENT
8947-000063/US

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicants: Hee-Kwan SON Group Art Unit: Unknown
Filing Date: December 17, 2003 Examiner: Unknown
Application No.: 10/736,832 Conf. No.: Unknown
Title: MONTGOMERY MODULAR MULTIPLIER AND
METHOD THEREOF USING CARRY SAVE
ADDITION

PRIORITY LETTER

February 25, 2004

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sirs:

Pursuant to the provisions of 35 U.S.C. 119, enclosed is/are a certified copy of the following priority document(s).

<u>Application No.</u>	<u>Date Filed</u>	<u>Country</u>
2003-26482	4/25/2003	Korea

In support of Applicant's priority claim, please enter this document into the file.

Respectfully submitted,

HARNESS, DICKEY, & PIERCE, P.L.C.

By  34,313
John A. Castellano, Reg. No. 35,094

P.O. Box 8910
Reston, Virginia 20195
(703) 668-8000

JAC/cah
Enclosure: 1 certified copy



별첨 사본은 아래 출원의 원본과 동일함을 증명함.

This is to certify that the following application annexed hereto is a true copy from the records of the Korean Intellectual Property Office.

출원번호 : 10-2003-0026482
Application Number

출원년월일 : 2003년 04월 25일
Date of Application APR 25, 2003

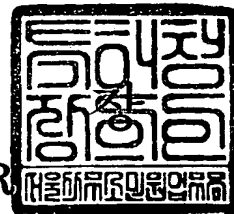
출원인 : 삼성전자주식회사
Applicant(s) SAMSUNG ELECTRONICS CO., LTD.



2003 년 09 월 16 일

특 허 청

COMMISSIONER



【서지사항】

【서류명】	명세서 등 보정서
【수신처】	특허청장
【제출일자】	2003.06.16
【제출인】	
【명칭】	삼성전자 주식회사
【출원인코드】	1-1998-104271-3
【사건과의 관계】	출원인
【대리인】	
【성명】	임창현
【대리인코드】	9-1998-000386-5
【포괄위임등록번호】	1999-007368-2
【대리인】	
【성명】	권혁수
【대리인코드】	9-1999-000370-4
【포괄위임등록번호】	1999-056971-6
【사건의 표시】	
【출원번호】	10-2003-0026482
【출원일자】	2003.04.25
【심사청구일자】	2003.04.25
【발명의 명칭】	모듈러 곱셈을 수행하는 연산장치
【제출원인】	
【접수번호】	1-1-2003-0149091-34
【접수일자】	2003.04.25
【보정할 서류】	명세서등
【보정할 사항】	
【보정대상항목】	별지와 같음
【보정방법】	별지와 같음
【보정내용】	별지와 같음
【취지】	특허법시행규칙 제13조·실용신안법시행규칙 제8조의 규 정에의하여 위와 같 이 제출합니다. 대리인 임창현 (인) 대리인 권혁수 (인)

【수수료】

【보정료】 0 원


【추가심사청구료】 0 원

【기타 수수료】 0 원

【합계】 0 원

【첨부서류】

1. 보정내용을 증명하는 서류_1통



1020030026482

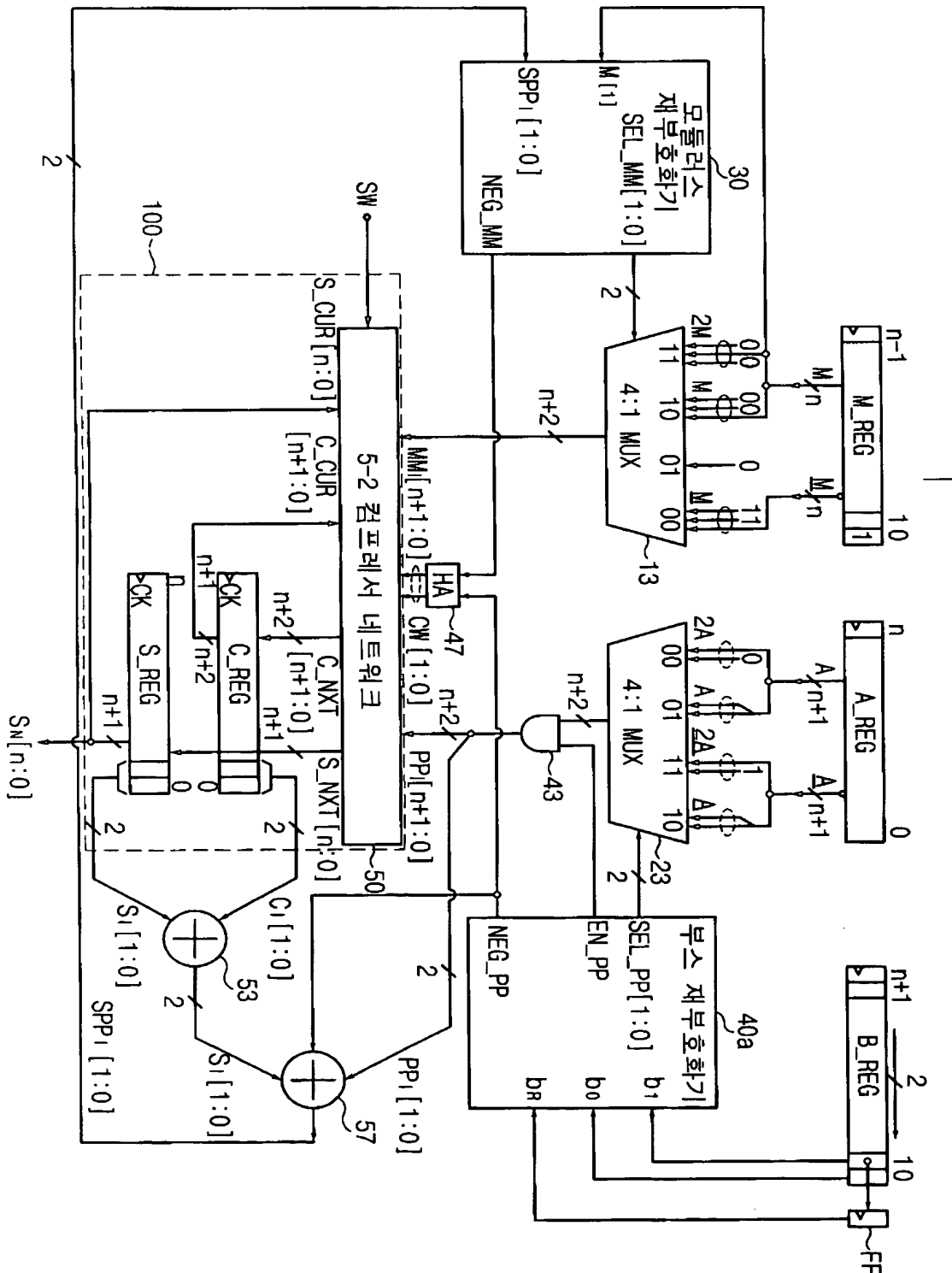
출력 일자: 2003/9/19

【보정대상항목】 도 8

【보정방법】 정정

【보정내용】

【도 8】





1020030026482

출력 일자: 2003/9/19

【서지사항】

【서류명】	특허출원서		
【권리구분】	특허		
【수신처】	특허청장		
【참조번호】	0001		
【제출일자】	2003.04.25		
【발명의 명칭】	모듈러 곱셈을 수행하는 연산장치		
【발명의 영문명칭】	MODULAR MULTIPLICATION CIRCUIT		
【출원인】			
【명칭】	삼성전자 주식회사		
【출원인코드】	1-1998-104271-3		
【대리인】			
【성명】	임창현		
【대리인코드】	9-1998-000386-5		
【포괄위임등록번호】	1999-007368-2		
【대리인】			
【성명】	권혁수		
【대리인코드】	9-1999-000370-4		
【포괄위임등록번호】	1999-056971-6		
【발명자】			
【성명의 국문표기】	손희관		
【성명의 영문표기】	SON, HEE-KWAN		
【주민등록번호】	690311-1551517		
【우편번호】	442-706		
【주소】	경기도 수원시 팔달구 망포동 동수원엘지빌리지 106동 1101호		
【국적】	KR		
【심사청구】	청구		
【취지】	특허법 제42조의 규정에 의한 출원, 특허법 제60조의 규정에 의한 출원심사를 청구합니다. 대리인 임창현 (인) 대리인 권혁수 (인)		
【수수료】			
【기본출원료】	20	면	29,000 원
【가산출원료】	33	면	33,000 원



1020030026482

출력 일자: 2003/9/19

【우선권주장료】	0	건	0	원
【심사청구료】	10	항	429,000	원
【합계】	491,000			원
【첨부서류】	1.	요약서·명세서(도면)_1통		

**【요약서】****【요약】**

본 발명에 따른 연산장치는, 제1피연산자, 제2피연산자 및 제3피연산자를 각각 저장하는 레지스터들; 상기 제1피연산자의 1의 보수값을 포함하는 제1연산값들 중 하나를 선택하는 제1멀티플렉서; 상기 제2피연산자의 1의 보수값을 포함하는 제2연산값들 중 하나를 선택하는 제2멀티플렉서; 상기 제1멀티플렉서를 제어하며, 상기 제1연산값에 포함된 상기 1의 보수값을 2의 보수값으로 변환하기 위한 제1보상비트를 발생하는 제1선택회로; 상기 제2멀티플렉서를 제어하며, 상기 제2연산값에 포함된 상기 1의 보수값을 2의 보수값으로 변환하기 위한 제2보상비트를 발생하는 제2선택회로; 그리고 상기 제1 및 제2선택회로의 출력값들과 상기 제1 및 제2보상비트를 입력하여 소정의 결과값을 구하는 누산기를 구비한다.

【대표도】

도 8

【색인어】

공개키, 모듈러 연산, 캐리 세이프



【명세서】

【발명의 명칭】

모듈러 곱셈을 수행하는 연산장치{MODULAR MULTIPLICATION CIRCUIT}

【도면의 간단한 설명】

도 1은 공개키 암호/복호화 알고리즘의 체계를 보여 준다.

도 2는 본 발명의 곱셈 연산장치에서 사용되는 캐리-세이브 방식의 컴프레서로서 전가산기들로 구성된 구조를 보여 준다.

도 3은 본 발명의 곱셈 연산장치에서 사용되는 캐리-세이브 방식의 컴프레서로서 반가산기를 포함하여 구성된 구조도이다.

도 4는 도 2 및 도 3의 컴프레서들로 구성되어 본 발명에 따른 곱셈장치에 적용될 누산기 구조의 실시예를 보여 준다.

도 5는 도 4에 보인 누산기의 상세회로도이다.

도 6은 도 2 및 도 3의 컴프레서들로 구성되어 본 발명에 따른 곱셈장치에 적용될 누산기 구조의 다른 실시예를 보여준다.

도 7은 도 6에 보인 누산기의 상세회로도이다.

도 8은 도 6에 보인 누산기를 채용한 본 발명에 따른 곱셈장치의 실시예를 보여 준다.

도 9은 도 6에 보인 누산기를 채용한 본 발명에 따른 곱셈장치의 다른 실시예를 보여 준다.

도 10은 도 8 또는 도 9의 곱셈장치에서 사용되는 모듈러스레코더의 회로도이다.

도 11은 도 6에서 사용되는 부스레코더의 회로도이다.



도 12는 도 9에서 사용되는 부스레코더의 회로도이다.

본 발명에 따른 도면들에서 실질적으로 동일한 구성과 기능을 가진 구성요소들에 대하여는 동일한 참조부호를 사용한다.

【발명의 상세한 설명】

【발명의 목적】

【발명이 속하는 기술분야 및 그 분야의 종래기술】

- <14> 본 발명은 데이터 보안을 위한 암호화(cryptography) 시스템에 관한 것으로서, 특히 암호화시스템에서의 모듈러 곱셈연산을 위한 장치 및 방법에 관한 것이다. 본 발명에서는, 최소한의 회로면적으로써 연산속도를 증대시키는 모듈러 곱셈장치를 제공한다.
- <15> 컴퓨터 네트워크 또는 유무선통신을 통하여 각종 데이터를 주고받는 정보통신 환경에서는 데이터의 보안성을 유지하기 위한 암호화 시스템의 중요성이 날로 커지고 있다. 특히, 전자결제 또는 인증시스템에서는 부호화(encryption) 및 복호화(decryption) 기술을 적용하여 보안성을 확보하는 것이 필수적이다. 암호화 기술은 크게 비밀키(대칭키 또는 공통키; secret key, symmetric key, private key, common key) 방식과 공개키(비대칭키; public key, asymmetric key) 방식으로 분류할 수 있다.
- <16> 비밀키 방식은 미합중국 상무성의 국립표준국(NBS)에 의한 DES(Data Encryption System) 암호 알고리즘이 대표적인 예이며, 그 외에 구 소련의 GOST(GOvernment STandard), 스위스의 IDEA(International Data Encryption Algorithm) 등이 있다. 비밀키 암호화 방식에서는 정보 교환 당사자간에 동일한 비밀키를 공유하여야 하므로 별도의 안전한 채널이 필요하고 특히 여



러 사람과의 정보교환시에는 한 사용자가 많은 비밀키 채널을 유지 및 관리하여야 한다는 단점이 있다.

<17> 반면에, 공개키 암호화 방식은 암호화하는 키와 복호화하는 키가 서로 다르기 때문에 둘 중 하나를 알더라도 그 에 대응하는 키를 알기 어렵도록 하는 시스템이다. 공개키 암호화 방식에서는, 하나의 비밀키와 다른 하나의 공개키를 사용하여 정보를 스크램블(scramble)/디스크램블(descramble)한다. 공개키 암호화 방식은, 키를 공유하기 위한 별도의 채널이 필요치 않아 키 관리가 수월하고 비밀키 방식에 비해 보안성은 향상되지만, 두가지의 서로 다른 키를 생성하고 이를 통하여 정보를 복원하기 위한 수학적 연산과정이 복잡하여 고속처리에 제한이 있는 것으로 알려져 있다.

<18> 공개키 암호화 알고리즘은, 1976년에 Diffie와 Helman에 의해 최초로 제안된 이래, 현재는 RSA(R.L.Rivest, A.Shamir, and L.Adelman; "A method for obtaining digital signatures and public-key cryptosystems"; Commun. ACM, Vol.21, pp.120-126, Feb. 1978, 또는 미합중국 특허 4,405,829) 알고리즘이 대표적인 방식으로 알려져 있다. 이 RSA방식은 다른 종류의 공개키 암호화 시스템인 Diffie-Hellman 방식 또는 DSA(Digital Signature Algorithm) 방식과 마찬가지로 모듈러 곱셈(modular multiplication)을 기본연산으로 하고 있다.

<19> RSA 알고리즘은 특히 1024비트 이상의 큰 정수(integer)를 기반으로 한 모듈로 연산에 의해 수행되며, 큰 정수 계수(modulus)의 소인수 분해가 매우 어려움에 그 안전성의 근거를 두고 있다.

<20> RSA 암호/복호화 알고리즘은 모듈러 멍승 연산(modular exponentiation operation)을 기본 연산으로 사용하고 있다. 도 1을 참조하면, 평문(plaintext) P를 암호문(ciphertext) C로 암호화(encryption)하기 위하여 두개의 양의 정수로 구성된 공개키 E(암호화지수 또는 암호화키)



및 M (계수;modulus)을 이용하여 연산식 $C = P^E \bmod M$ 에 의해 암호문 C 를 생성한다. 암호문 C 를 평문 P 로 복호화(decryption)함에 있어서는 비밀키 D (복호화지수)를 이용하여 연산식 $P = C^D \bmod M$ (C 를 D 번 곱한 값을 M 으로 나눈 나머지를 취함)에 의해 평문 P 를 구한다. 즉, 앨리스(Alice)가 밥(Bob)에게 암호화된 메시지를 보내기 위해서는, 모든 사람들에게 공개되어 있는 밥의 공개키 E 로써 메시지를 암호화한 뒤 공개채널(unsecure channel)을 통하여 밥에게 메시지(암호문 C)를 보낸다. 그러면, 밥은 자신만이 알고 있는 비밀키 D 를 사용해서 암호화된 메시지를 복호화해서 원래의 메시지(평문 P)를 읽게 된다.

<21> 암호/복호화를 위한 키를 선택하기 위해서는, 우선 두개의 매우 큰 소수(prime number) p 및 q 를 임의로 선택하여 $M = p \times q$ 를 계산하고, $\gcd[D, (p-1)(q-1)] = 1$ 을 만족하는 수를 복호화 키 D 로 한다. 암호화 키 E 는 $E \times D = 1 \bmod [(p-1)(q-1)]$ 을 만족하는 암호화키 E 를 구한다.

<22> RSA를 위한 모듈로연산(modular multiplication)은 내부에 곱셈과 나눗셈이 복합되어 있어 계산구조가 복잡하고 워드 크기가 통상 1024 비트 이상으로 크기 때문에 고속의 실시간 처리를 위한 하드웨어 모듈의 구현이 어렵다. 즉, 사용하는 키 값이 증가하면 모듈러 역승을 위한 연산속도가 감소되므로 고속연산을 위해서는 역승에 필요한 모듈러 곱셈 수를 가능한 최소로 줄여야 한다. RSA 알고리즘의 주된 연산인 모듈로 역승은 연속된 모듈로 곱셈으로 구성되므로 RSA 암호화 시스템 구현의 관건은 모듈로 곱셈기의 설계에 있다.

<23> RSA 암호화 시스템을 구현하는 방법은 크게 두가지로 나눌 수 있다. 하나는 어레이 곱셈기(array multiplier)를 응용한 것이고, 다른 하나는 1985년에 제안된 몽고메리(Montgomery) 알고리즘(P.L.Montgomery, "Modular Multiplication Without Trial Division", Mathematics of Computation, vol.44, No.170, pp.519-521, 1985)을 이용한 곱셈방법이다.



<24> 몽고메리 알고리즘은 하드웨어 구현이 어려운 임의의 수에 대한 모듈러 연산을 단순히 곱셈 및 덧셈연산과 쉬프트(shift)연산으로 변환하여 해결하기 때문에 디지털 IC설계를 통한 구현이나 중앙처리장치(CPU) 또는 디지털신호처리장치(DSP) 등에서의 알고리즘 실현이 용이하도록 하는 장점이 있다. 몽고메리 알고리즘에서는 곱셈연산의 전후에 피연산자 변환(operand transformation) 과정이 필요하다. 따라서, 단일 곱셈연산에서는 다른 일반적인 모듈러곱셈기에 비하여 처리성능이 느릴 수 있지만, RSA 알고리즘과 같이 동일한 모듈러스에 대하여 반복적인 곱셈연산을 행하는 응용영역에서는 매번의 피연산자 변환작업이 필요하지 않기 때문에 다른 모듈러 곱셈기에 비해 고속으로 암호화 작업을 처리할 수 있다.

<25> 곱셈연산의 속도를 높이는 방법 중의 하나가 래디스(radix; 기수)의 값을 늘리는 것이다. 래디스 값이 증가하면 곱셈연산과정에서 수행되는 누산(accumulation)의 반복횟수가 그만큼 줄어든다. 그러나, 래디스 값이 증가하면 한 번의 누산을 위해 필요한 과정이 그만큼 복잡해지고 한번의 누산에 소모되는 시간이 그만큼 증가한다. 예를 들면, 래디스-2 연산 알고리즘은 하드웨어로 구현하는 것이 용이하지만 래디스-4 연산 알고리즘에 비하여 누산의 반복횟수가 2배가 된다. 그러나, 래디스-4 연산방식은 기본적으로 래디스-2 연산알고리즘에 비하여 하드웨어 구현이 복잡하고 각 반복수행의 지연시간이 증가하는 약점이 있다.

<26> 따라서, 하드웨어의 크기를 늘리거나 더 복잡하게 하지 않으면서 모듈러연산의 속도를 증가시킬 수 있도록 하는 적절한 설계가 중요하다.

【발명이 이루고자 하는 기술적 과제】

<27> 따라서, 본 발명의 목적은 하드웨어의 크기를 늘리지 않으면서 모듈러 연산속도를 향상시키는 곱셈 연산장치를 제공함에 있다.



- <28> 본 발명의 다른 목적은 래딕스-4 논리연산을 기반으로 하여 몽고메리 곱셈 알고리즘의 하드웨어 구성 및 동작을 효율적으로 실현하는 곱셈 연산장치를 제공함에 있다.
- <29> 본 발명에 따른 연산장치는, 제1피연산자, 제2피연산자 및 제3피연산자를 각각 저장하는 레지스터들; 상기 제1피연산자의 1의 보수값을 포함하는 제1연산값들 중 하나를 선택하는 제1멀티플렉서; 상기 제2피연산자의 1의 보수값을 포함하는 제2연산값들 중 하나를 선택하는 제2멀티플렉서; 상기 제1멀티플렉서를 제어하며, 상기 제1연산값에 포함된 상기 1의 보수값을 2의 보수값으로 변환하기 위한 제1보상비트를 발생하는 제1선택회로; 상기 제2멀티플렉서를 제어하며, 상기 제2연산값에 포함된 상기 1의 보수값을 2의 보수값으로 변환하기 위한 제2보상비트를 발생하는 제2선택회로; 그리고 상기 제1 및 제2선택회로의 출력값들과 상기 제1 및 제2보상비트를 입력하여 소정의 결과값을 구하는 누산기를 구비한다.
- <30> 상기 누산기는, 제1캐리연산모드와 제2캐리연산모드에서 선택적으로 동작하도록 설계되며, 상기 제1연산자의 비트길이가 n 인 경우에 $n+3$ 의 비트길이를 처리하도록 구성된다. 제1캐리연산모드와 제2캐리연산모드에서 선택적으로 동작하도록 하기 위하여, 상기 누산기의 비트위치마다 배치되어 소정의 제어신호에 응답하여 상기 제1 및 제2캐리연산모드사이에서 절환되는 소정 갯수의 멀티플렉서들이 포함된다.
- <31> 상기 제1선택회로는, 상기 제1피연산자의 하위 2번째 비트와, 상기 캐리워드 및 상기 섹 워드의 합으로 이루어진 워드의 하위 2비트와 상기 제2연산값의 하위 2비트를 합산한 값을 입력하고 그 값들의 조합논리연산을 통하여 상기 제1멀티플렉서를 제어하는 신호를 발생한다.
- <32> 상기 제2선택회로는, 상기 제3피연산자의 하위 2비트와, 상기 제3피연산자의 2비트 우측 쉬프트된 비트를 입력한다.



- <33> 상기 제2선택회로로부터 발생하는 제어신호에 응답하여 상기 제2멀티플렉서로부터 제공되는 선택된 상기 제2연산값이 상기 누산기로 전달되는 경로를 통제하는 게이트가 본 발명의 연산장치에 더 포함된다.
- <34> 본 발명의 다른 실시예에 있어서, 상기 제2멀티플렉서는, 상기 제2선택회로로부터 제공되는 제어신호에 응답하여 상기 제2연산값들 중 상기 제2피연산자의 1배수값 및 2배수값 중 하나를 선택하는 제1유닛; 상기 제2선택회로로부터 제공되는 상기 제어신호에 응답하여 상기 제2연산자의 상기 1의 보수로 된 값의 제1배수값 및 제2배수값 중 하나를 선택하는 제2유닛; 그리고 상기 제2선택회로로부터 제공되는 제어신호에 응답하여 상기 제1 및 제2유닛으로부터 제공된 값들 중 하나를 선택하는 제3유닛으로 구성된다.

【발명의 구성 및 작용】

- <35> 본 발명의 실시예에서는, 래디스-4 논리연산을 기반으로 하여 몽고메리 곱셈알고리즘을 적용하며, 이를 래디스-4 인터리브 몽고메리 곱셈 알고리즘(Radix-4 Interleaved Montgomery Multiplication Algorithm)이라 하고 R4IMM로 약칭한다.
- <36> 본 발명의 실시예에서 보이는 곱셈장치의 논리연산체계는 공개키 방식의 암호화알고리즘을 적용하는 컴퓨터 시스템 또는 통신망에 적용될 수 있으며, 또한 휴대가능한 집적회로 카드(또는 스마트카드)에 내장되어 운용될 수 있다.
- <37> 본 발명에 따른 실시예는 적어도 1024비트 이상의 큰 정수를 기반으로 하는 모듈로연산에 적용될 수 있다.
- <38> 본 발명의 실시예를 설명하기에 앞서, 본 발명에 적용되는 모듈러 곱셈 알고리즘 R4IMM에서 참조되는 패라미터들(parameters)을 다음과 같이 정의한다.



- <39> M은 모듈러스(modulus)로서 2보다 큰 양의 정수(integer)로서 홀수값(예컨대, 3, 7 등)을 가진다.
- <40> M'는 조건식 $(-M * M') \bmod 4 = 1$ 을 만족하는 정수이다.
- <41> n은 모듈러스 M의 비트길이(bit length)를 나타내는 값으로서 양의 정수이다. 본 발명의 실시예에서는 8로 할 것이다.
- <42> N은 모듈러스 M의 디지트길이(digit length)를 나타내는 값으로서 양의 정수이다.
- <43> 여기서, n은 N의 2배값($n = 2N$)으로서, n이 8이라면 N은 4가 된다.
- <44> R은 $R = 2^n = 2^{2N}$ 의 조건을 만족하는 양의 정수이다. n이 8이라면 R은 256이 된다.
- <45> R^{-1} 은, R의 모듈로 역곱수로서, 조건식 $(R * R^{-1}) \bmod M = 1$ 을 만족하는 양의 정수이다.
- <46> A는, 피승수(multiplicand)로서, $0 \leq A < M$ 의 조건을 만족하는 정수이다.
- <47> B는, 승수(multiplier)로서, $0 \leq B < M$ 의 조건을 만족하는 정수이다. 여기서, $B = \sum_{i=0}^{N-1} b_i A^i$, $b_i \in \{0, 1, 2, 3\}$ 이다. b_i 는 2비트(1디지트)이며 승수를 구성하는 각 디지트를 나타낸다.
- <48> 본 발명에 적용되는 R4IMM 알고리즘은 다음과 같다.
- <49> $S_0 := 0$
- <50> for I := 0 to (N-1)
- <51> $q_I := (((S_I + b_I A) \bmod 4) * M') \bmod 4$
- <52> $S_{I+1} := (S_I + b_I A + q_I M) / 4$
- <53> endfor



<54> if ($S_N \geq M$) $S_N := S_N - M$

<55> R4IMM 알고리즘에서 I 는 디지털 인덱스 또는 연산의 반복회수를 나타내며, 알고리즘을 수행한 총 곱의 결과는 $S_N = A \cdot B \cdot R^{-1} \pmod{M}$ 으로 나타난다. 총 곱 S_N 의 범위는 $0 \leq S_N < M$ 이다. 상기 알고리즘에서 몫 q_I 는 '이전까지 누산기(accumulator)에 저장된 값 S_I 에 부분곱(partial product) $b_I A$ (이하, PP_I : partial product)를 더하여 만든 값 $S_I + b_I A (= b_I A + PP_I)$ 의 하위 2비트가 "00"이 될 수 있도록 더해 줄 M 의 갯수'를 나타낸다. 잉여수 체계(residue number system; RNS)에서 어떤 수에 계수 M 의 정수배를 다한 수는 원래의 수와 같으므로, 계수 M 의 정수배인 모듈러스곱 $q_I M$ (이하, MM_I ; multiple of modulus)을 더한 수는 원래의 수와 같다. 또한, $S_I + b_I A$ 의 하위 2비트를 "00"으로 만든 후에 래디스값 4로 나누면(즉, 2비트씩 오른쪽으로 쉬프트함) 유효자리의 숫자는 그대로 보존되기 때문에 정보가 유실되지 않는다.

<56> 이와 같은 R4IMM 알고리즘을 하드웨어적으로 구현하기 위해서는, 부분곱 PP_I 와 모듈러스곱 MM_I 를 구해야 한다. 단위승수 b_I 와 몫 q_I 가 2비트(즉, 1디지털)이므로, 부분곱 PP_I 와 모듈러스곱 MM_I 는 다음과 같이 4가지 경우의 가능값들을 각각 가지도록 설정될 수 있을 것이다(여기서, $b_I \in \{0, 1, 2, 3\}$ 이고 $q_I \in \{0, 1, 2, 3\}$ 으로 함).

<57> $b_I A = PP_I \in \{0, A, 2A, 3A\}$

<58> $q_I M = MM_I \in \{0, M, 2M, 3M\}$ 식 1



<59> 그러나, 위 [식 1]과 같이 부분곱 PP_I 와 모듈러스곱 MM_I 를 설정하게 되면, 값 $3A$ 와 $3M$ 를 계산할 때 A 또는 M 을 1비트 쉬프트한 값과 원래 값의 합을 구해야 한다($2A+A$, $2M+M$). 이를 하드웨어로 구현하기 위해서는 이러한 값을 계산할 독립된 가산기(adder)를 사용하거나 또는 이 값들을 미리 계산하여 메모리 등에 저장해 놓고 필요할 때마다 참조하는 방법을 사용하여야 한다. 그러한 방법은 하드웨어의 낭비를 초래하고, 또한 그 값들($3A$, $3M$)을 구하는 시간(미리 계산해 두거나 실시간으로 계산함)을 추가적으로 고려하여 하드웨어를 설계하여야 하므로 성능 저하의 요인이 될 수 있다.

<60> 그리하여, 본 발명에서는, 부분곱 PP_I 와 모듈러스곱 MM_I 를 구함에 따른 하드웨어 부담을 줄이고자 한다.

<61> 먼저, 부분곱 PP_I 값 $3A$ 를 구함에 따른 하드웨어적인 부담을 줄이기 위하여, 래딕스-4 부스(Booth) 재부호화(recoding) 방법을 사용하여 부분곱 PP_I 의 가능값들을 아래의 식 2와 같이 전환하여 줌으로써, 피승수 A 의 반전(bit-inversion) 및 쉬프트(bit-shifting)만으로 해당하는 부분곱 PP_I 의 값들을 구한다. 즉, 승수 B 의 단위비트들을 2비트씩(즉, 1디지트씩) 재부호화한 후에 이를 피승수 A 와 결합시켜 부분곱 PP_I 를 생성하는 산술 연산 방식이다.

<62>
$$b_I A = PP_I \in \{-2A, -A, 0, +A, +2A\} \quad \text{식 2}$$

<63> 승수 B 의 워드크기(즉, 길이 또는 비트수)가 n 이고 이진표기로 " $b_{n-1}b_{n-2}..b_1b_0$ "로 표시될 때 래딕스-4 부스 재부호화에 의한 부분곱 PP_I 는 다음의 [표 1]과 같이 정리된다.



<64> 【표 1】

b_{2I+1}	b_{2I}	b_{2I-1}	PP_I
0	0	0	0
0	0	1	+A
0	1	0	+A
0	1	1	+2A
1	0	0	-2A
1	0	1	-A
1	1	0	-A
1	1	1	0

<65> 래딕스-4 부스 레코딩에서는, 승수 B로부터 2비트 단위로 재부호화한 다음에 이를 피승수 A와 곱하기 때문에, n비트의 승수 B에 대하여 n/2개의 부분곱이 생성된다. 즉, 부분곱의 행의 수가 1/2로 줄어들어 따라, [표 1]에 보인 바와 같이, b_I 에 대한 인덱스가 b_{2I+1} , b_{2I} 및 b_{2I-1} 로 표기된다. 또한, "-A"와 "-2A"는 각각 "A"와 "2A"의 2의 보수(2's complement)로 된다. 부스 재부호화를 통해 부분곱 PP_I 의 값들을 [식 1] 또는 [표 1]에 보인 바와 같이 2의 보수 형식으로 설정한 이유는, 비트반전 및 비트쉬프트만으로 부분곱 PP_I 의 값들을 구할 수 있기 때문에 [식 1]에 보인 "3A"를 구하기 위하여 추가적인 하드웨어를 사용하여야 하는 부담을 줄이기 위함이다.

<66> 다음으로, 본 발명에 따라 모듈러스곱 MM_I 를 구하는 과정을 설명한다.

<67> R4IMM 알고리즘에서 원하는 모듈러스곱 MM_I (즉, $q_I M$)는 $S_I + PP_I$ (이하, 부분누산합; SPP_I)의 하위 2비트를 "00"으로 만들 수 있는 모듈러스 M의 정수배에 해당하는 값이다. 그리고, 모듈러스 M은 홀수이어야 하므로 모듈러스 M의 최하위 비트 $M[0]$ 는 항상 "1"이 된다.

<68> 그러므로, 부분누산합 SPP_I 의 하위 2비트 $SPP_I[1:0]$ 과 모듈러스 M의 하위 두번째 비트 $M[1]$ 을 이용하여 아래의 [표 2]와 같이 모듈러스곱 MM_I 의 값을 결정할 수 있다.



<69> 【표 2】

$SPP_I[1]$	$SPP_I[0]$	$M[1] \ (M[0] = 1)$	MM_I
0	0	0	0 (즉, $q_I = 0$)
0	0	1	0 (즉, $q_I = 0$)
0	1	0	$-M$ (즉, $q_I = -1$)
0	1	1	$+M$ (즉, $q_I = +1$)
1	0	0	$+2M$ (즉, $q_I = +2$)
1	0	1	$+2M$ (즉, $q_I = +2$)
1	1	0	$+M$ (즉, $q_I = +1$)
1	1	1	$-M$ (즉, $q_I = -1$)

<70> [표 2]로부터 MM_I 의 가능값들은 다음과 같음을 알 수 있다.

<71> $MM_I \in \{-M, 0, +M, +2M\}$ 식 3

<72> [표 2]에서 " $-M$ "은 " M "의 2의 보수를 취한 값이다. 따라서, 전술한 부분곱 PP_I 에 대한 경우와 같은 이유로, [식 1]에 보인 값 " $3M$ "을 구할 필요가 없음을 알 수 있다.

<73> 래딕스-4 부스 재부호화(recoding) 기법을 사용하기 위해서는 피승수 A와 승수 B가 모두 2의 보수 형식으로 표현되어야 하고 길이가 부호비트(sign bit)를 포함하여 짝수이어야 한다.

<74> 본 발명에 따른 R4IMM 알고리즘에서 입력값이 되는 피승수 A 및 승수 B와 출력값이 되는 S_N 의 범위는 다음과 같다.

<75> $-M \leq A, B, S_N < M$ 식 4

<76> 원래의 몽고메리 알고리즘에서는 "for" 루프연산을 거쳐 처리된 출력값 S_N 은 $0 \leq S_N < 2M$ 의 범위를 가지며 입력값이 되는 피승수 A 및 승수 B 는 $0 \leq A, B < M$ 의 범위로 제한된다. 그리하여, 모듈러 멍승(exponentiation; 또는 모듈러 지수화)을 위하여 출력값 S_N 을 다음 모듈러 곱 연산의 입력으로 바로 사용하지 못하고 S_N 이 M 보다 큰 경우 S_N 에서 M 을 빼주어야 한다($S_N - M$; 후감산(post-reduction)이라고 함).

<77> 그러나, 본 발명에 따른 R4IMM 알고리즘에서는, 식 4에 보인 바와 같이 입력값 A 및 B 와 출력값 S_N 의 범위가 동일하기 때문에($0 \leq A, B, S_N < M$), 매 모듈러 곱셈연산후 후감산 과정($S_N \geq M$ 일 때, $S_N - M$)을 거치지 않고 출력값 S_N 을 다음 곱셈의 입력으로 바로 사용할 수 있다. 그리고, 모듈러 멍승연산이 모두 종료된 후에 얻어지는 결과값의 부호비트를 검사하여 부호비트가 "1"이면(즉, 결과값이 음수이면) M 을 더하여 최종 결과값이 0과 M 사이의 값으로 되도록 한번만 교정해 주면 된다.

<78> 2의 보수 체계에서 부분곱 PP_I 및 모듈러스곱 MM_I 의 가능값들 " $-2A$ ", " $-A$ " 및 " $-M$ "을 생성하기 위해서는, 비트쉬프트(1-bit left shift)와 비트반전(bit-inversion)에 의해 1의 보수(1's complement)인 " $2A$ ", " A " 및 " M "을 생성한 후 "1"을 더해 주어야 한다. 만약 "1"을 더하기 위하여 캐리 전파 가산기(carry propagation adder)를 사용한다면 이 과정에서 발생하는 캐리 전파 지연시간(carry propagation delay)에 의해 전체 하드웨어의 성능이 나빠지므로 이는 바람직하지 않다.

- <79> 본 발명의 실시예에서는, 부분곱 PP_I 가 피승수 A 와 부호가 다른 경우(즉, $PP_I = -A$ 또는 $-2A$ 인 경우)와 모듈러스곱 MM_I 가 M 과 부호가 다른 경우(즉, $MM_I = -M$ 인 경우)에, 비트반전과 비트쉬프트만으로 " $2A$ ", " A " 및 " M "의 1의 보수들($\underline{2A}$, \underline{A} , \underline{M})을 생성하고 2의 보수를 위한 +1의 보상값들을 모아서 독립된 워드(이하, 보상워드 CW_I)를 만들어 덧셈에 참여시킨다.
- <80> 2의 보수화를 위한 부분곱 PP_I 와 모듈러스곱 MM_I 의 비트반전(negation) 필요 여부에 따른 보상워드 CW_I 의 가능값들은 다음과 같다.
- <81> $CW_I \in \{0, 1, 2\}$ 식 5
- <82> 식 5에서, $CW_I = 0$ ("00")이면, 해당하는 PP_I 와 MM_I 의 모두에 대하여 비트반전이 필요하지 않은 경우이다. $CW_I = 1$ ("01")이면, 해당하는 PP_I 와 MM_I 중 어느 하나의 값에 대하여 비트반전이 필요한 경우이다. $CW_I = 2$ ("10")이면, 해당하는 PP_I 와 MM_I 의 모두에 대하여 비트반전이 필요한 경우이다.
- <83> 매우 긴 길이의 정수(예컨대, 1024 비트 이상)를 덧셈할 때에는 캐리의 전파지연시간을 줄이는 것이 중요하다. 이를 위하여, 본 발명의 실시예에서는 캐리-세이브 가산기(carry-save adder)를 사용한다. 캐리-세이브 가산기는 캐리 전파 가산기(carry propagation adder)와 달리 가산기의 길이와 상관없이 일정한 지연시간을 갖고 그 지연시간 자체도 짧기 때문에 동작속도가 빠르다.
- <84> 캐리 세이브 가산기를 이용하여 곱셈기를 구현하기 위해서는, 전술한 R4IMM 알고리즘에서 부분합 S_I 가 2개의 워드, 즉 캐리워드(carry word; C_I)와 섬워드(sum word; S_I)의 합으로



변환되어야 한다(예컨대, "0011" = "0010"(C_I) + "0001"(S_I)) . 따라서, R4IMM 알고리즘에서 매 디지털을 처리할 때마다 누산기에서의 새로운 값을 구하는 식은 다음과 같이 정리된다.

<85>
$$(C_{I+1} + S_{I+1}) = [(C_I + S_I) + PP_I + MM_I + CW_I] / 4 \quad \text{식 6}$$

<86> [식 6]에서 우변은 입력값에 해당하고 좌변은 출력값에 해당한다. 즉, 캐리 세이브 가산기에서, C_I, S_I, PP_I, MM_I 및 CW_I로 되는 5개의 워드가 입력값들이며, C_{I+1} 및 S_{I+1}로 되는 2개의 워드가 출력값들이 된다. 다시 말하면, [식 6]은 본 발명에 따른 R4IMM 알고리즘을 하드웨어적으로 실현하기 위한 곱셈장치의 덧셈연산을 위한 단위 입출력구성을 나타낸다.

<87> [식 6]에 따른 덧셈연산을 처리하기 위한 장치의 단위구조의 일례는, 도 2에 보인 바와 같이, 3개의 전가산기(full adder; FA)들로 구성된 5-입력/2-출력 컴프레서(5-2 compressor; 또는 5:2 카운터)로 구현될 수 있다.

<88> 도 2에 보인 5-2 컴프레서는, 5개의 주입력들(primary inputs) PI₁, PI₂, PI₃, PI₄ 및 PI₅와 2개의 부입력들(secondary inputs) SI₁ 및 SI₂에 대하여 2개의 주출력들(primary outputs) PO₁ 및 PO₂와 2개의 부출력들(secondary outputs) SO₁ 및 SO₂를 발생하도록 설계되어 있다. 5-2 컴프레서의 입출력간의 관계는 다음과 같다.

<89>
$$(PI_1+PI_2+PI_3+PI_4+PI_5)+(SI_1+SI_2) = (2 \cdot PO_1+PO_2)+(2 \cdot SO_1+2 \cdot SO_2) \quad \text{식 7}$$

- <90> 5-2 컴프레서의 연결을 통하여 누산기를 구성하면 한 자리에 위치한 5-2 컴프레서의 부출력 SO_1 과 SO_2 는 자신과 인접한 상위자리 5-2 컴프레서의 부입력 SI_1 과 SI_2 에 각각 연결된다. 그리고, 주입력들에 더해야 할 값들이 인가되고 덧셈의 결과가 주출력들에 나타난다. 또한, 가장 낮은 자리에 위치한 5-2 컴프레서의 부입력들에 인가되는 값은 각각 "0"이고 가장 높은 자리에 위치한 컴프레서의 부출력들은 사용하지 않는다.
- <91> 5-2 컴프레서의 5개의 주입력 터미널들 $PI_1 \sim PI_5$ 를 더해야 할 5개의 워드들에 할당하는 방법은 다양하다. 본 발명의 실시예에서는, 부분곱 PP_I 와 모듈러스곱 MM_I 가 궤환입력된 캐리워드 C_I 및 섬워드 S_I 보다 복잡한 과정을 거쳐서 만들어 지므로 5-2 컴프레서 내부에서 전달경로가 보다 짧은 주입력터미널인 PI_4 및 PI_5 에 PP_I 및 MM_I 를 각각 배정한다.
- <92> 도 2에서 5-2 컴프레서의 구성요소로 사용한 전가산기 FA의 불리언 논리식은 다음과 같다.
- <93>
$$CO = AI \cdot BI + BI \cdot CI + CI \cdot AI$$
- <94>
$$SO = AI \oplus BI \oplus CI$$
- <95> 한편, 보상워드 CW_I 는 2비트로써 충분히 나타낼 수 있으므로($CW_I \in \{0, 1, 2\}$), 전체 누산기 구성에서 하위 두개 자리 제외한 상위 자리의 컴프레서들은 도 3에 보인 바와 같이, 도 2의 구조에서 상단의 전가산기 FA1을 반가산기 HA1(half adder)로 대체하고 CW_I 를 위한 주입력 터미널 PI_3 을 제거한 형태의 축약된 5-2 컴프레서를 사용할 수 있다. 이렇게 함으로써 전체적인 하드웨어의 크기를 좀 더 줄일 수 있다.



<96> 도 3에서 축약된 5-2 컴프레서의 구성요소로 사용한 반가산기 HA의 불리언 논리식은 다음과 같다.

<97> $CO = AI \cdot BI$

<98> $SO = AI \oplus BI$

<99> 도 2 및 도 3에 보인 컴프레서를 이용한 누산기의 구조는 후술하는 누산기의 전체 구성도(도 4 내지 도 7)를 통하여 보다 상세히 설명할 것이다. 도 2 또는 도 3에서 부입력들 SI_1 및 SI_2 는 도 4 및 도 6의 누산기 구조에서 입력캐리들 CI_1 및 CI_2 에 해당하며, 부출력들 SO_1 및 SO_2 는 출력캐리들 CO_1 및 CO_2 에 해당한다.

<100> 통상적으로, 몽고메리 모듈러 곱셈기에서 운용되는 RSA 알고리즘에서는 암호화하려는 평문이나 복호화하려는 암호문은 모두 무부호(부호가 없음)로 된 워드이다. 하지만, 본 발명에 따른 R4IMM 알고리즘에서는 래딕스-4 부스 재부호화 방식을 이용하고 있으므로, 무부호 워드를 2의 보수로 확장하여야 하고 확장된 길이는 짝수값이어야 한다. 또한, 전술한 식 2 및 식 3으로부터 알 수 있는 바와 같이, 부분곱 PP_1 와 모듈러스곱 MM_1 가 피승수 A와 모듈러스 M의 2배까지 가능하므로("+2A", "-2A", "+2M"), 곱셈기의 크기가 그 값들을 연산하는 과정에서 비트범람(bit overflow)이 발생하지 않도록 충분히 커야 한다. 하지만, 너무 크게 설계하면 하드웨어의 소모 및 전력의 소모가 발생하므로 곱셈기의 크기를 필요한 만큼의 적절한 크기로 결정하여야 한다.



- <101> 모듈러스 M 의 비트길이가 n 인 경우, 곱셈기가 수용하여야 할 워드의 크기는 부분곱 PP_I 와 모듈러스곱 MM_I 의 부호비트를 포함하여 $n+2$ 비트의 길이를 기본적으로 포함하여야 한다. 여기서, 추가된 2비트는, 입력 피연산자 A 와 M 의 2배값("2M", "-2A", "2M")을 포함하고 2의 보수체계에서의 부호를 포함하여야 하기 때문이다. 또한, 현재의 누산된 값 C_I 및 S_I 에 부분곱 PP_I , 모듈러스곱 MM_I , 그리고 보상워드 CW_I 를 더한 결과를 수용하기 위해서는(즉, 비트범람(bit overflow)을 방지하기 위해서는) 1비트를 더 확장하여야 한다.
- <102> 따라서, n -비트의 길이를 가지는 모듈러스 M 에 대하여, 본 발명에서 적용될 누산기가 처리하는 최소한의 비트길이는 $n+3$ 이다.
- <103> 도 4는, 모듈러스 M 의 길이가 n 비트인 경우에, 도 2 및 도 3에 보인 5-2 컴프레서들로 구성된 컴프레서 네트워크 45를 포함한 누산기 구조의 일례를 보여 준다. 도 5는 도 4에 보인 컴프레서 네트워크 45의 상세회로를 보인 것으로서, 도 4와 동시에 참조될 것이다.
- <104> 도 4에서 처리되는 총 비트 수는 $n+3$ 이며, 각 비트에 해당하는 컴프레서들 $FCPh_0$, FCP_1 , $HCP_2 \sim HCP_{n+2}$ 는 캐리-세이프 방식으로 연결 배치된다.
- <105> 도 4에 보인 컴프레서들은 기본적으로 5-2 컴프레서의 구조를 채용하고 있으나, 사용되는 보상워드 CW_I 의 입력 필요성과 관련하여 전술한 도 2 내지 도 3에 설명한 바와 같이 입출력 구조의 효율적인 변경을 적용하고 있다. 즉, 하위 2비트 $[1:0]$ 에 해당하는 컴프레서들 FCP_1 및 $FCPh_0$ 에만 2비트의 보상워드 $CW_I[1:0]$ 가 입력값에 포함되며, 컴프레서들 FCP_1 및 $FCPh_0$ 는 도 2에 보인 5-2 컴프레서의 입출력구조로 되어 있다. 여기서, 하위 첫번째 비트에 해당하는 컴프레서 $FCPh_0$ 는 도 2에 보인 전가산기들의 구조에서 하단의 전가산기(FA3)가 포함되지 않은



5-2 컴프레서의 구조로 되어 있다. 나머지 상위 비트들 $[n+2:2]$ 에 해당하는 컴프레서들 $HCP_{n+2} \sim HCP_2$ 는 전술한 도 3에 보인 컴프레서의 입출력 구조로 되어 있다.

<106> 또한, 컴프레서들로부터 발생하는 캐리워드들($n+2$ 비트) 과 섬워드($n+1$ 비트)들을 저장하기 위한 캐리 레지스터 C_REG와 섬 레지스터 S_REG가 제공된다. 캐리레지스터 C_REG 및 섬 레지스터 S_REG에는 각 비트에 해당하는 플립플롭들 FF가 포함된다. 캐리 레지스터 C_REG 및 섬 레지스터 S_REG에 포함된 플립플롭들 FF의 출력들은 단일워드로 된 최종곱셈의 결과 $SN[n:0]$ 을 구하기 위하여 캐리 전과 가산기 CPA로 제공되는 한편 매 누산과정마다 필요한 "2비트 우측 쉬프트"연산을 수행하기 위하여 컴프레서의 하위 비트 입력으로 제공된다. 즉, 각 비트워치의 컴프레서에서 발생하는 캐리워드는 1비트 하위에 위치한 컴프레서로 제공되며, 각 비트워치의 컴프레서에서 발생하는 섬워드는 2비트 하위에 위치한 컴프레서로 제공된다.

<107> 도 4를 참조하면, 첫번째 비트워치로부터 각 컴프레서에서 발생하는 출력캐리들 CO_1 및 CO_2 는 다음 단의 컴프레서의 입력캐리 CI_1 및 CI_2 로 제공된다. 여기서, 출력캐리들 CO_1 및 CO_2 는 도 2 또는 도 3에서의 부출력들 SO_1 및 SO_2 에 해당하며, 입력캐리들 CI_1 및 CI_2 는 도 2 또는 도 3에서의 부입력들 SO_1 및 SO_2 에 해당한다. 첫번째 비트워치의 컴프레서 $FCPh_0$ 의 입력은 $CW_1[0]$, $MM_1[0]$ 및 $PP_1[0]$ 과 두번째 비트워치의 컴프레서 FCP_1 로부터 제공되는 캐리워드 $C_1[0]$ 과 세번째 비트워치의 컴프레서 HCP_2 로부터 제공되는 섬워드 $S_1[0]$ 이다. 두번째 비트워치의 컴프레서 FCP_1 의 입력은 $CW_1[1]$, $MM_1[1]$ 및 $PP_1[1]$ 과 세번째 비트워치의 컴프레서 HCP_2 로부터 제공되는 캐리워드 $C_1[1]$ 과 네번째 비트워치의 컴프레서 HCP_3 으로부터 제공되는 섬워드 $S_1[2]$ 이다. 두번째 비트워치의 컴프레서 FCP_1 부터 $n+2$ 번째 비트워치의 컴프레서 HCP_{n+1} 까지,



동일한 방식으로 입출력 연결구조로 되어 있다. 단, 부호비트로서 최상위 비트위치에 해당하는 컴프레서 HCP_{n+2} 에는 자체의 캐리워드 $C_I[n+1]$ 과 섬워드 $S_I[n]$ 이 캐환입력된다.

<108> 도 4에서는 컴프레서 네트워크 45에 의한 최종적인 곱셈의 결과가 단일 워드가 아닌 캐리워드(C)와 섬워드(S)로 구분되어 나타난다. 따라서, 단일워드의 이진수로 된 최종 곱셈의 결과 $S_N[n:0]$ 을 얻기 위해서는 캐리워드(C)와 섬워드(S)의 값을 캐리-전파 가산기(carry-propagation adder) CPA를 통하여 캐리워드(C)와 섬워드(S)를 더해 주어야 한다. 캐리-전파 가산기는 여러가지가 있지만 가장 간단한 형태가 리플캐리가산기(ripple carry adder)로서, 도 4에 보인 바와 같이 전가산기들 FA의 출력캐리와 입력캐리를 직렬로 연결하여 구성할 수 있다. 캐리-전파 가산기 CPA를 통하여 단일워드로 된 결과값 $S_N[n:0]$ 은 각 비트에 해당하는 플립플롭들 FF로 된 레지스터 F_REG를 통하여 출력된다.

<109> 도 6은 도 4에 컴프레서네트워크 45와는 달리 최종적인 곱셈의 결과인 $S_N[n:0]$ 을 산출하는 컴프레서 네트워크 50을 포함한 누산기의 다른 실시예(100)를 보여 준다. 도 7은 도 6에 보인 컴프레서네트워크 50의 상세회로를 보인 것으로서, 도 6과 함께 참조될 것이다.

<110> 도 6에 보인 컴프레서들 $FCPh_0$, FCP_1 , $HCP'_2 \sim HCP'_{n+2}$ 도 전술한 도 4의 경우와 마찬가지로 기본적으로 5-2 컴프레서의 구조로 되어 있으나, 하드웨어의 크기를 더욱 작게 하기 위하여, 하위 2비트($FCPh_0$, FCP_1)를 제외한 각 비트 위치의 컴프레서들 $HCP'_2 \sim HCP'_{n+2}$ 의 구조가 도 4(또는 도 5)에 보인 것과 다른 구조를 가진다.

<111> 즉, 캐리워드와 섬워드를 합산하기 위한 별도의 독립적인 캐리-전파 가산기(도 4 또는 도 5의 CPA)와 그 합산된 결과를 저장하기 위한 별도의 레지스터(도 4 또는 도 5의 F-REG)를 사용하는 대신에, 3개의 2:1 멀티플렉서들 MUX로 이루어진 멀티플렉서 그룹들 $MXG_0 \sim MXG_{n+1}$ 이

하위 2비트를 제외한 각 비트 위치에 배치된다. 각 비트위치에서 상기 멀티플렉서 그룹의 위치는 도 2에서 중단의 전가산기 FA2와 하단의 전가산기 FA3의 사이에 해당한다. 멀티플렉서 그룹들 $MXG_0 \sim MXG_{n+1}$ 은 제어신호 SW에 응답하여 컴프레서 네트워크 50의 동작구조를 선택적으로 운용하기 위하여 제공된다.

<112> 3개의 2:1 멀티플렉서가 차지하는 면적은 1개의 전가산기(Full-Adder; FA)와 1개의 플립 플롭(Flip-Flop; FF)이 차지하는 면적의 절반 정도이므로, 캐리워드와 섬워드를 합산하고 저장하기 위한 별도의 캐리전파가산기와 레지스터를 사용하는 경우보다 하드웨어의 구조적인 크기를 줄일 수 있다.

<113> 또한, 도 6 또는 도 7에 보인 구조는 도 4(또는 도 5)에 보인 캐리워드 및 섬워드들의 연결관계와는 다르게 되어 있다. 즉, 캐리워드가, 도 4에서는 하위의 비트위치로만 제한되는데 반해, 하위의 비트위치는 물론 상위 비트위치의 컴프레서의 멀티플렉서 그룹에도 제공된다. 또한, 각 컴프레서로부터 발생하는 출력캐리 CO_1 및 CO_2 는 다음 상위비트위치의 컴프레서의 입력캐리 CI_1 및 CI_2 로서 각 컴프레서의 멀티플렉서 그룹 MXG으로 직접 입력된다. 한편, 섬워드는 하위 2비트위치에는 물론 상위 1비트위치의 컴프레서의 멀티플렉서 그룹에도 제공된다.

<114> 즉, 첫번째 및 두번째의 비트위치의 컴프레서 $FCPh_0$ 및 FCP_1 의 입력들은 도 4(또는 도 5)의 경우와 동일하다.

<115> 세번째 비트위치의 컴프레서 HCP'_2 의 입력은, $MM_1[2]$, $PP_1[2]$, 네번째 비트위치의 컴프레서 HCP'_3 으로부터 제공되는 캐리워드 $C_1[2]$, 다섯번째 비트위치의 컴프레서 HCP'_4 로부터 제공되는 섬워드 $S_1[2]$, 두번째 비트위치의 컴프레서 FCP_1 으로부

터 제공되는 캐리워드 $C_I[0]$, 그리고 자체적으로 발생된 섬워드 $S_I[0]$ 이다. 여기서, 이전 하위비트위치에서 캐환되는 $C_I[0]$ 및 $S_I[0]$ 는 해당하는 멀티플렉서 그룹 MXG_0 에 입력된다.

또한, 멀티플렉서 그룹 MXG_0 에 인가되는 입력 "0"은 캐리 전파 가산기의 가장 낮은 자리에 있는 전가산기의 캐리입력이다.

<116> 세번째 비트위치의 컴프레서 HCP'_2 부터 $n+2$ 번째 비트위치의 컴프레서 HCP'_{n+1} 에 이르기까지, 동일한 방식으로 입출력 연결구조로 되어 있다. 단, 부호비트로서 최상위 비트위치에 해당하는 컴프레서 HCP'_{n+2} 에는 도 4(또는 도 5)와 같이 자체의 캐리워드 $C_I[n+1]$ 과 섬워드 $S_I[n]$ 이 캐환입력된다.

<117> 5-2 컴프레서 네트워크 50내에서 멀티플렉서 그룹들 $MXG_0 \sim MXG_{n+1}$ 을 이용한 이 같은 연결구조는 캐리-세이브 방식의 연산과 캐리-전파 방식의 연산을 선택적으로 수행하기 위함이다.

<118> 도 6 또는 도 7을 참조하면, 하위비트 $[1:0]$ 를 제외한 비트위치들에서 중단의 전가산기와 하단의 전가산기사이 3개씩의 2:1 멀티플렉서들(MUX)이 배치된다. 멀티플렉서들(MUX)은 제어신호 SW에 응답하여 누산기가 선택적으로 캐리-세이브 방식으로 동작하거나 캐리-전파 방식으로 동작하도록 한다. 제어신호 SW의 상태에 따라, 전체 곱셈과정은, 매 클럭주기마다 1디지트(2비트)를 처리하여 캐리워드와 섬워드를 구하는 주연산과정과, 주연산과정에서 얻어진 캐리워드와 섬워드를 합산하여 최종 결과값 $S_N[n:0]$ 를 구하는 후변환과정으로 나뉘어 진다.

<119> 주연산과정에서는, 제어신호 SW가 "0"으로 설정되어, 전술한 도 4(또는 도 5)와 마찬가지로 캐리-세이브 모드로 동작하며 $N+1$ 번의 루프(loop)를 돌면서 매 클럭사이클마다 1디지트(2비트) 단위로 처리되고 생성된 캐리워드와 섬워드 각각 레지스터 C_REG 및 S_REG에 저장된다. 이후의 후변환과정에서는, 제어신호 SW가 "1"로 설정됨에 따라, 각 비트위치의 하단에 위치한

전가산기들이 직렬로 연결되어 캐리-전파 모드로 동작한다. 후변환과정에서는 이전에 주연산 과정에서 캐리레지스터 C_REG와 섬레지스터 S_REG에 각각 저장되어 있던 캐리워드와 섬워드의 더해진 결과가 다시 섬레지스터 S_REG에 저장된다.

- <120> 비트길이가 매우 큰 경우(예컨대, 1024비트 이상)에는 후변환과정을 한 클럭사이클동안에 완료하는 것이 불가능하다. 따라서, 이 경우에는 후변환과정이 시작하는 시점부터 클럭을 비활성상태로 만들고 캐리-전파 모드에 의한 연산이 완료된 후에 클럭을 다시 활성화시켜 섬레지스터 S_REG에 최종결과값 $S_N[n:0]$ 을 저장한다.
- <121> 도 8은 도 6에 보인 누산기 100을 채용하여 본 발명에 다른 R4IMM 모듈러 곱셈연산을 수행하기 위한 모듈러곱셈기의 전체적인 구성을 보여 준다.
- <122> 도 8의 모듈러곱셈기는 모듈러스 레지스터 M_REG, 피승수 레지스터 A_REG, 승수 레지스터 B_REG, 캐리 레지스터 C_REG, 섬 레지스터 S_REG, 4:1 멀티플렉서들 13 및 23, 모듈러스 재부호화기 30, 부스 재부호화기 40a, 앤드게이트 43, 반가산기 47, 5-2 컴프레서 네트워크 50, 그리고 덧셈기들 53 및 57을 포함한다. 5-2 컴프레서 네트워크 50과 캐리 레지스터 C_REG 및 섬 레지스터 S_REG는 전술한 도 6 또는 7에 보인 누산기 100을 구성한다.
- <123> 캐리 레지스터 C_REG 및 섬 레지스터 S_REG는 도 4 또는 도 6에 보인 것과 동일하며, 레지스터들 M_REG, A_REG, C_REG 및 S_REG는 병렬 입출력형(PIPO; Parallel-In parallel-Out) 레지스터들이다. 레지스터들 M_REG, A_REG, C_REG 및 S_REG는 $N+1(= (n/2)+1)$ 회의 반복연산이 진행되어 최종 결과값 $S_N[n:0]$ 이 산출될 때까지 클럭 CK의 매 사이클에 응답하여 동작한다.
- <124> 모듈러스 레지스터 M_REG는 n비트로 된 모듈러스 $M[n-1:0]$ 을 저장한다. 모듈러스 M은 항상 홀수이므로 모듈러스 레지스터 M_REG의 마지막 비트위치(LSB)는 항상 "1"로 설정되어 있

다. 또한, 모듈러스 M 은 항상 양수이므로 부호를 저장할 필요가 없다. 피승수 레지스터 A_REG 는 $n+1$ 비트의 길이로 되어 있으며 피승수 $A[n:0]$ 를 저장한다. 피승수 레지스터 A_REG 의 길이가 $n+1$ 인 것은 피승수 A 가 양수와 음수 모두 가능하므로 부호비트(n 번째 비트)를 포함시켜야 하기 때문이다. 승수 레지스터 B_REG 는 병렬-입력/직렬-출력(PISO; Parallel-In Serial-Out) 레지스터이며 승수 B 를 저장한다. 승수 B 에는 부호비트와 짝수의 비트길이를 맞추기 위한 비트가 포함되어야 하므로, 승수 레지스터 B_REG 는 $n+2$ 의 비트길이를 되어 있으며 n 번째 및 $n+1$ 번째 비트가 부호를 나타낸다. 승수 레지스터 B_REG 는 매 클럭사이클마다 2비트씩 우측 쉬프트동작을 수행한다. 승수 레지스터 B_REG 는 매 클럭사이클마다 우측 쉬프트후에 하위 2비트 b_1 및 b_0 와 비트 b_R (비트 b_R 은 이전 사이클의 비트 b_1) 부스 재부호화기 40으로 제공한다.

<125> 모듈러스 레지스터 M_REG 로부터 제공되는 모듈러스 M 과 모듈러스 M 의 1의 보수값 \overline{M} 을 포함하여 전술한 식 3에 보인 모듈러스곱 MM_I 의 값들(\overline{M} , 0, M , $2M$)이 4:1 멀티플렉서 13에 인가된다. 한편, 피승수 레지스터 A_REG 로부터 제공되는 피승수 A 과 피승수 A 의 1의 보수값 \overline{A} 를 포함하여 전술한 식 2에 보인 부분곱 PP_I 의 값들($2\overline{A}$, \overline{A} , 0, A , $2A$)이 4:1 멀티플렉서 23에 인가된다. 멀티플렉서들 13 및 23에 전달된 모듈러스곱 및 부분곱의 값들은 1의 보수값들(\overline{M} , \overline{A} , $2\overline{A}$)을 포함하고 있으며, 멀티플렉서들 13 및 23에 의해 선택된 후에 누산기 50에서 보상워드 CW_I 를 이용한 연산을 통하여 2의 보수화될 것이다.

<126> 모듈러스 재부호화기 30은, 모듈러스곱 MM_I 의 값을 선택하기 위한 조합논리회로로서, 전술한 표2와 관련하여 설명한 바와 같이 모듈러스 레지스터 M_REG 로부터 하위 2번째 비트 $M[1]$ 과 부분누산합의 하위 2비트 $SPP_I[1:0]$ 을 입력한 다음, 멀티플렉서 13에 전송된 모듈러스곱 MM_I 의 값들 중 하나를 선택하기 위한 신호 $SEL_MM[1:0]$ 를 발생한다. 모듈러스 재부호화기 30

은 또한, 전술한 보상워드 CW_I 와 관련하여 설명한 바와 같이, 선택된 모듈러스곱 MM_I 의 비트반전 여부를 알려주기 위한 신호 NEG_MM 을 발생한다. 아래의 표 3은 모듈러스 재부호화기 30의 입출력 상관관계와 그에 따라 멀티플렉서 13에 입력된 모듈러스곱 MM_I 의 값들(\underline{M} , 0, M , $2M$) 중에서 선택되는 모듈러스곱 MM_I 의 값을 나타낸다.

<127> 【표 3】

입 력			출 력		선택된 MM_I [$n+1:0$]
$SPP_I[1]$	$SPP_I[0]$	$M[1]$	$SEL_MM[1:0]$	NEG_MM	
0	0	0	11	0	0
0	0	1	11	0	0
0	1	0	01	1	\underline{M}
0	1	1	00	0	\underline{M}
1	0	0	10	0	$2M$
1	0	1	10	0	$2M$
1	1	0	00	0	M
1	1	1	01	1	\underline{M}

<128> 모듈러스 재부호화기 30에 의해 선택된 모듈러스곱 MM_I 의 값은 모듈러스 M 의 2배수를 위한 비트와 부호비트를 포함하는 $n+2$ 비트의 길이를 가지며 5-2 컴프레서네트워크 50으로 입력된다.

<129> 모듈러스 재부호화기 30의 상세회로 구성을 보여주는 도 10을 참조하면, 모듈러스 재부호화기 30은 인버터들 $INV1 \sim INV3$ 과 낸드게이트들 $ND1 \sim ND7$ 을 포함하여 이루어진다. 낸드게이트 $ND1$ 의 입력은 $SPP_I[1]$ 및 $M[1]$ 의 반전비트이다. 낸드게이트 $ND2$ 의 입력은 $SPP_I[1]$ 및 $SPP_I[0]$ 의 반전비트이다. 낸드게이트 $ND3$ 의 입력은 $SPP_I[1]$ 의 반전비트와 $SPP_I[0]$ 및 $M[1]$ 이다. 낸드게이트 $ND4$ 의 입력은 $SPP_I[1]$ 과 $SPP_I[0]$ 및 $M[1]$ 이다. 낸드게이트 $ND5$ 의 입력은 $SPP_I[1]$ 의 반전비트와 $SPP_I[0]$ 및 $M[1]$ 의 반전비트이다. 낸드게이트 $ND6$ 의 입력은 낸드게이트들 $ND1 \sim ND3$

의 출력들이다. 낸드게이트 ND7의 입력은 낸드게이트들 ND4 및 ND5의 출력들이다. 멀티플렉서 13에 인가되는 제어신호 SEL_MM[1:0] 중 상위비트 SEL_MM[1]은 낸드게이트 ND6으로부터, 하위비트 SEL_MM[0]은 SPPI[0]의 반전비트로 제공된다. 반가산기 47에 인가되는 제어신호 NEG_MM은 낸드게이트 ND7로부터 출력된다.

<130> 다시 도 8을 참조하면, 부스 재부호화기 40a는, 부분곱 PP_I 의 값을 선택하기 위한 조합 논리회로로서, 전술한 표1과 관련하여 설명한 바와 같이 승수 레지스터 B_REG로부터 하위 2비트 b_1 및 b_0 와 이전 싸이클의 비트 b_1 인 b_R 을 입력받은 다음, 멀티플렉서 23에 전송된 부분곱 PP_I 의 값들 중 하나를 선택하기 위한 신호 SEL_PP[1:0]를 발생한다. 우측 쉬프트된 비트 b_R 은 플립플롭 FF를 통하여 부스 재부호화기 40a로 공급된다. 부스 재부호화기 40a는, 또한, 전술한 보상워드 CW_I 와 관련하여 설명한 바와 같이, 선택된 부분곱 PP_I 의 비트반전 여부를 결정하기 위한 신호 NEG_PP를 발생한다. 신호 NEG_PP는 모듈러스 재부호화기 30로부터 제공되는 신호 NEG_MM과 함께 반가산기 47로 인가되어 2비트의 보상워드 $CW_I[1:0]$ 가 컴프레서네트워크 50으로 제공되도록 한다.

<131> 도 8에 사용된 부스 재부호화기 40a의 상세회로 구성을 보여주는 도 11을 참조하면, 부스 재부호화기 40a는 인버터들 INV4~INV6과 낸드게이트들 ND8~ND17을 포함하여 구성된다. 낸드게이트 ND8의 입력은 b_0 및 b_R 이다. 낸드게이트 ND9의 입력은 b_1 및 b_0 이다. 낸드게이트 ND10의 입력은 b_1 및 b_R 이다. 낸드게이트 ND11의 입력은 b_0 및 b_R 의 반전비트들이다. 낸드게이트 ND12의 입력은 b_R 의 반전비트와 b_0 이다.



낸드게이트 ND13의 입력은 b_0 의 반전비트와 b_1 이다 . 낸드게이트 ND14의 입력은 b_R 의 반전비트와 b_1 이다. 낸드게이트 ND15의 입력은 낸드게이트들 ND8~ND10의 출력들이다. 낸드게이트 ND16의 입력은 낸드게이트들 ND11~ND13의 출력들이다. 낸드게이트 ND17의 입력은 낸드게이트들 ND13 및 ND14의 출력들이다. 멀티플렉서 23에 인가되는 제어신호 SEL_PP[1:0]의 상위비트 SEL_PP[1]은 b_1 과 동일하며 하위비트 SEL_PP[0]은 낸드게이트 ND15로부터 출력된다. 도 8의 앤드게이트 43에 인가되는 제어신호 EN_PP는 낸드게이트 ND16으로부터 출력된다. 도 8의 반가산기 47에 인가되는 제어신호 NEG_PP는 낸드게이트 ND17로부터 출력된다.

<132> 다시 도 8을 참조하면, 한편, 멀티플렉서 23으로부터 부스 재부호화기 40a의 선택신호 SEL_PP[1:0]에 의해 선택된 부분곱 PP_I 의 값은 $n+2$ 비트 길이로서 앤드게이트 43에 제공된다. 이 때, 앤드게이트 43에는 또한 부스재부호화기 40a으로부터 제공되는 제어신호 EN_PP가 인가된다. 승수비트들 b_1 , b_0 및 b_R 이 모두 "0"이거나 "1"일 때에는 부분곱 PP_I 의 값이 0이므로, 이 경우에는 제어신호 EN_PP를 "0"으로 설정하여 앤드게이트 43의 출력이 항상 0으로 되도록 한다.

<133> 아래의 표 4는 부스 재부호화기 40a의 입출력 상관관계와 그에 따라 멀티플렉서 23과 앤드게이트 43에 의해 선택되는 부분곱 PP_I 의 값을 나타낸다.

<134>

【표 4】

입 력			출 력			선택된
b ₁	b ₀	b _R	SEL_PP[1:0]	EN_PP	NEG_PP	PP _I [n+1:0]
0	0	0	don't care	0	0	0
0	0	1	00	1	0	A
0	1	0	00	1	0	A
0	1	1	10	1	0	2A
1	0	0	11	1	1	<u>2A</u>
1	0	1	01	1	1	<u>A</u>
1	1	0	01	1	1	<u>A</u>
1	1	1	don't care	0	0	0

<135> 부스 재부호화기 40a로부터 제공되는 부분곱 선택신호 SEL_PP[1:0]와 게이트신호 EN_PP에 의해 선택된 부분곱 PP_I[n+1:0]의 값은 피승수 A의 2배수를 위한 비트와 부호비트를 포함하는 n+2 비트의 길이로서 컴프레서네트워크 50으로 입력된다.

<136> 누산기 100에서는, 도 6과 관련하여 설명한 바와 같이, 캐리-세이브 모드로 동작하여 캐리워드와 섬워드를 산출하는 주연산과정과, 캐리레지스터 C_REG 및 섬레지스터 S_REG에 각각 저장된 캐리워드와 섬워드를 합산하여 그 결과를 섬레지스터 S_REG에 저장하는 후변환과정(캐리-전파 모드)을 통하여 최종값 S_N[n:0]이 생성된다. 캐리-세이브 동작모드의 주연산과정에서 캐리전파모드의 후변환과정으로의 전환은, 전술한 도 6과 관련하여 설명한 바와 같이, 제어신호 SW가 "0"에서 "1"로 천이됨에 따라 각 비트위치의 2:1 멀티플렉서들을 통하여 전가산기들이 직렬로 연결됨에 의해 진행된다.

<137> 한편, 모듈러스곱 MM_I의 선택에 참여하는 부분누산합의 하위 2비트 SPP_I[1:0]를 만들기 위하여, 캐리레지스터 C_REG와 섬레지스터 S_REG로부터 추출된 하위 2비트 C_I[1:0] 및 S_I[1:0]



는 덧셈기 53을 통하여 합산된 다음($S_I[1:0]$), 그 합산된 결과인 2비트의 섬워드 $S_I[1:0]$ 는 다시 덧셈기 57에서 부분곱의 하위 2비트 $PP_I[1:0]$ 과 합산된다.

<138> 전체적인 연산과정의 이해를 돕기 위하여, 도 8에서, $S_CUR[n:0]$ 및 $C_CUR[n+1:0]$ 는 현재 처리되는 섬워드와 캐리워드를 각각 가리키며, $S_NXT[n:0]$ 및 $C_NXT[n+1:0]$ 는 $n+1$ 회의 반복 연산과정에서 다음에 처리될 섬워드와 캐리워드를 각각 가리킨다.

<139> 덧셈기 57로부터 생성된 2비트의 부분누산합 $SPP_I[1:0]$ 은 모듈러스 재부호화기 30으로 인가된다. 합산기들 53 및 57은 2비트 가산기로 구성된다.

<140> 도 8에서 도 6 또는 도 7에 보인 구조에 실질적으로 해당하는 부분은 컴프레서네트워크 50를 비롯하여 캐리레지스터 C_REG 및 섬레지스터 S_REG 와 합산기 53를 포함함을 이해하여야 한다.

<141> 도 9은 도 8에 보인 모듈러곱셈기의 다른 실시예로서, 피승수 레지스터 A_REG 로부터 제공되는 1의 보수값을 포함한 부분곱 PPI 의 값들($2A$, A , $\underline{2A}$, \underline{A}) 중에서 하나를 선택하는 4:1 멀티플렉서 23을 3개의 2:1 멀티플렉서들 24, 26 및 28로 대체한 형태이다. 또한, 멀티플렉서가 분할됨에 따라, 도 8에 보인 부스 재부호화기 40a가, 멀티플렉서들 24 및 26을 제어하기 위한 신호 SFT_PP 를 발생하는 부스 재부호화기 40b로 대체된다.

<142> 도 12를 참조하면, 도 9의 부스 재부호화기 40b는 인버터들 $INV7 \sim INV9$ 와 낸드게이트들 $ND18 \sim ND26$ 을 포함하여 구성된다. 낸드게이트 $ND18$ 의 입력은 b_1 , b_0 의 반전비트, 그리고 b_R 의 반전비트이다. 낸드게이트 $ND19$ 의 입력은 b_1 의 반전비트, b_0 , 그리고 b_R 이다. 낸드게이트 $ND20$ 의 입력은 b_1 의 반전비트 및 b_R 이다. 낸드게이트 $ND21$ 의 입력은 b_0 및 b_R 의 반전비트이다. 낸드게이트 $ND22$ 의 입력은 b_0 의 반전비트와 b_1 이다. 낸드게이트 $ND23$ 의 입력



은 b_1 및 b_R 의 반전비트이다. 낸드게이트 ND24의 입력은 낸드게이트들 ND18~ND19의 출력들이다. 낸드게이트 ND25의 입력은 낸드게이트들 ND20~ND22의 출력들이다. 낸드게이트 ND26의 입력은 낸드게이트들 ND22 및 ND23의 출력들이다. 도 9의 멀티플렉서들 24 및 26에 인가되는 제어신호 SFT_PP는 낸드게이트 ND24로부터 출력된다. 도 9의 앤드게이트 43에 인가되는 제어신호 EN_PP는 낸드게이트 ND25로부터 출력된다. 도 9의 반가산기 47에 인가되는 제어신호 NEG_PP는 낸드게이트 ND26으로부터 출력된다. 도 9에 보인 2:1 멀티플렉서들 24 및 26은 도 12의 부스 재부호화기 40b에서 발생하는 제어신호 SFT_PP에 응답하여 동작하며 도 9의 2:1 멀티플렉서 28은 도 12의 부스 재부호화기 40b에서 발생하는 NEG_PP에 응답하여 동작한다.

<143> 아래의 표 5는 도 7의 부스 재부호화기 40의 입출력 상관관계와 그에 따라 멀티플렉서들 24, 26 및 28과 앤드게이트 43에 의해 선택되는 부분곱 PP_I 의 값들을 나타낸다.

<144> 【표 5】

입 력			출 력			선택된 $PP_I [n+1:0]$
b_1	b_0	b_R	SFT_PP	EN_PP	NEG_PP	
0	0	0	0	0	0	0
0	0	1	0	1	0	A
0	1	0	0	1	0	A
0	1	1	1	1	0	2A
1	0	0	1	1	1	<u>2A</u>
1	0	1	0	1	1	<u>A</u>
1	1	0	0	1	1	<u>A</u>
1	1	1	0	0	0	0

<145> 멀티플렉서들 24 및 26의 출력들은 다시 2:1 멀티플렉서 28로 입력된다. 멀티플렉서 28는 부스 재부호화기 40b에서 제공되는 제어신호 NEG_PP에 응답하여 최종적으로 하나의 값을 선택한다. 멀티플렉서 28에서는, 피승수 A의 배수값의 1의 보수값(

2A 또는 A)을 선택할 것인지 아니면 피승수 A의 배수값(2A 또는 A)를 선택할 것인지만을 결정하면 되므로, 비트반전의 여부를 결정하는 제어신호 NEG_PP에 의해 제어된다.

<146> 상술한 실시예에서 보인 본 발명의 수단 또는 방법에 준하여 본 발명의 기술분야에서 통상의 지식을 가진 자는 본 발명의 범위내에서 본 발명의 변형 및 응용이 가능하다. 예를 들면, 도 10 내지 도 12에 보인 모듈러스 재부호화기 및 부스 재부호화기의 회로들은 관련된 표 3 내지 표 5에 보인 결과에 맞도록 구성함에 있어서 다양한 논리 조합들이 가능할 것이다.

【발명의 효과】

<147> 전술한 본 발명의 실시예에 의하면, 본 발명은 모듈러 곱셈기에서 최소한의 구성요소만으로 캐리-세이프 모드와 캐리전파 모드로 선택적으로 동작할 수 있는 누산기의 구조를 제공함으로써, 하드웨어적인 부담을 줄이면서 모듈러 연산속도를 향상시키는 효과가 있다.

【특허청구범위】**【청구항 1】**

연산장치에 있어서:

제 1피연산자, 제2피연산자 및 제3피연산자를 각각 저장하는 레지스터들;

상기 제1피연산자의 1의 보수값을 포함하는 제1연산값들 중 하나를 선택하는 제1멀티플렉서;

상기 제2피연산자의 1의 보수값을 포함하는 제2연산값들 중 하나를 선택하는 제2멀티플렉서;

상기 제1멀티플렉서를 제어하며, 상기 제1연산값에 포함된 상기 1의 보수값을 2의 보수값으로 변환하기 위한 제1보상비트를 발생하는 제1선택회로;

상기 제2멀티플렉서를 제어하며, 상기 제2연산값에 포함된 상기 1의 보수값을 2의 보수값으로 변환하기 위한 제2보상비트를 발생하는 제2선택회로; 그리고

상기 제1 및 제2선택회로의 출력값들과 상기 제1 및 제2보상비트를 입력하여 소정의 결과값을 구하는 누산기를 구비함을 특징으로 하는 연산장치.

【청구항 2】

제1항에 있어서,

상기 누산기가, 제1캐리연산모드와 제2캐리연산모드에서 선택적으로 동작함을 특징으로 하는 연산장치.

**【청구항 3】**

제2항에 있어서,

상기 누산기가, 상기 제1피연산자의 비트길이가 n 인 경우에 $n+3$ 의 비트길이를 처리하도록 구성됨을 특징으로 하는 연산장치.

【청구항 4】

제2항에 있어서,

상기 누산기가, 비트위치마다 배치되어 소정의 제어신호에 응답하여 상기 제1 및 제2캐리연산모드사이에서 절환되는 소정 갯수의 멀티플렉서들을 구비함을 특징으로 하는 연산장치.

【청구항 5】

제2항에 있어서,

상기 누산기가, 캐리워드와 섬워드를 각각 저장하는 레지스터들을 구비함을 특징으로 하는 연산장치.

【청구항 6】

제5항에 있어서,

상기 제1선택회로가, 상기 제1피연산자의 하위 2번째 비트와, 상기 캐리워드 및 상기 섬워드의 합으로 이루어진 워드의 하위 2비트와 상기 제2연산값의 하위 2비트를 합산한 값을 입력함을 특징으로 하는 연산장치.



【청구항 7】

제1항에 있어서,

상기 제2선택회로가, 상기 제3피연산자의 하위 2비트와, 상기 제3피연산자의 2비트 우측 쉬프트된 비트를 입력함을 특징으로 하는 연산장치.

【청구항 8】

제1항에 있어서,

상기 제2선택회로로부터 발생하는 제어신호에 응답하여 상기 제2멀티플렉서로부터 제공되는 선택된 상기 제2연산값이 상기 누산기로 전달되는 경로를 통제하는 게이트를 더 구비함을 특징으로 하는 연산장치.

【청구항 9】

제1항에 있어서,

상기 제2멀티플렉서가:

상기 제2선택회로로부터 제공되는 제어신호에 응답하여 상기 제2연산값들 중 상기 제2연산자의 1배수값 및 2배수값 중 하나를 선택하는 제1유닛;

상기 제2선택회로로부터 제공되는 상기 제어신호에 응답하여 상기 제2연산자의 제1배수값 및 제2배수값의 상기 1의 보수로 된 값들 중 하나를 선택하는 제2유닛; 그리고

상기 제2선택회로로부터 제공되는 제어신호에 응답하여 상기 제1 및 제2유닛으로부터 제공된 값들 중 하나를 선택하는 제3유닛을 구비함을 특징으로 하는 연산장치.



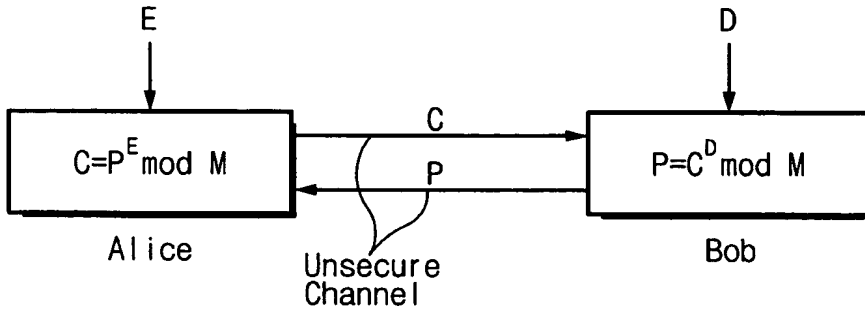
【청구항 10】

제9항에 있어서,

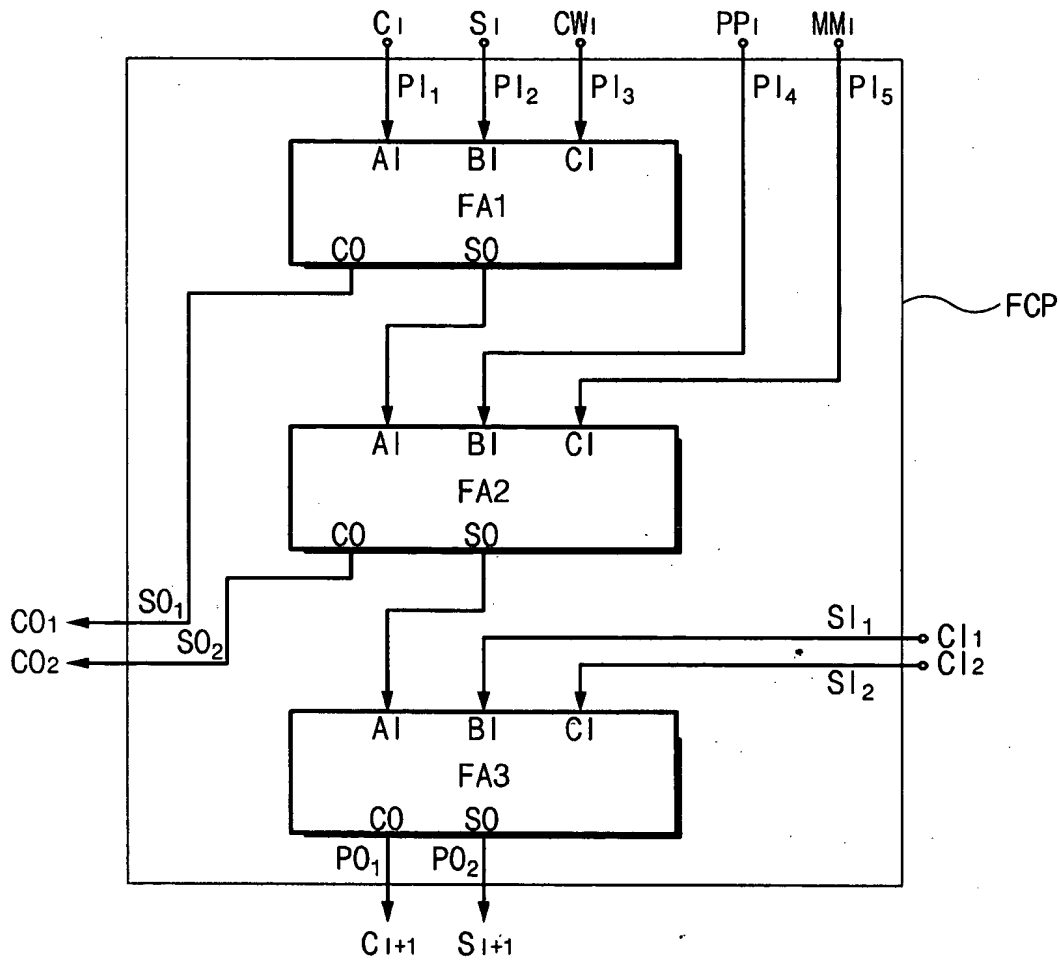
상기 제2선택회로로부터 발생하는 제어신호에 응답하여 상기 제3유닛으로부터 제공되는
선택된 상기 제2연산값이 게이트를 통하여 상기 누산기로 전달됨을 특징으로 하는 연산장치.

【도면】

【도 1】

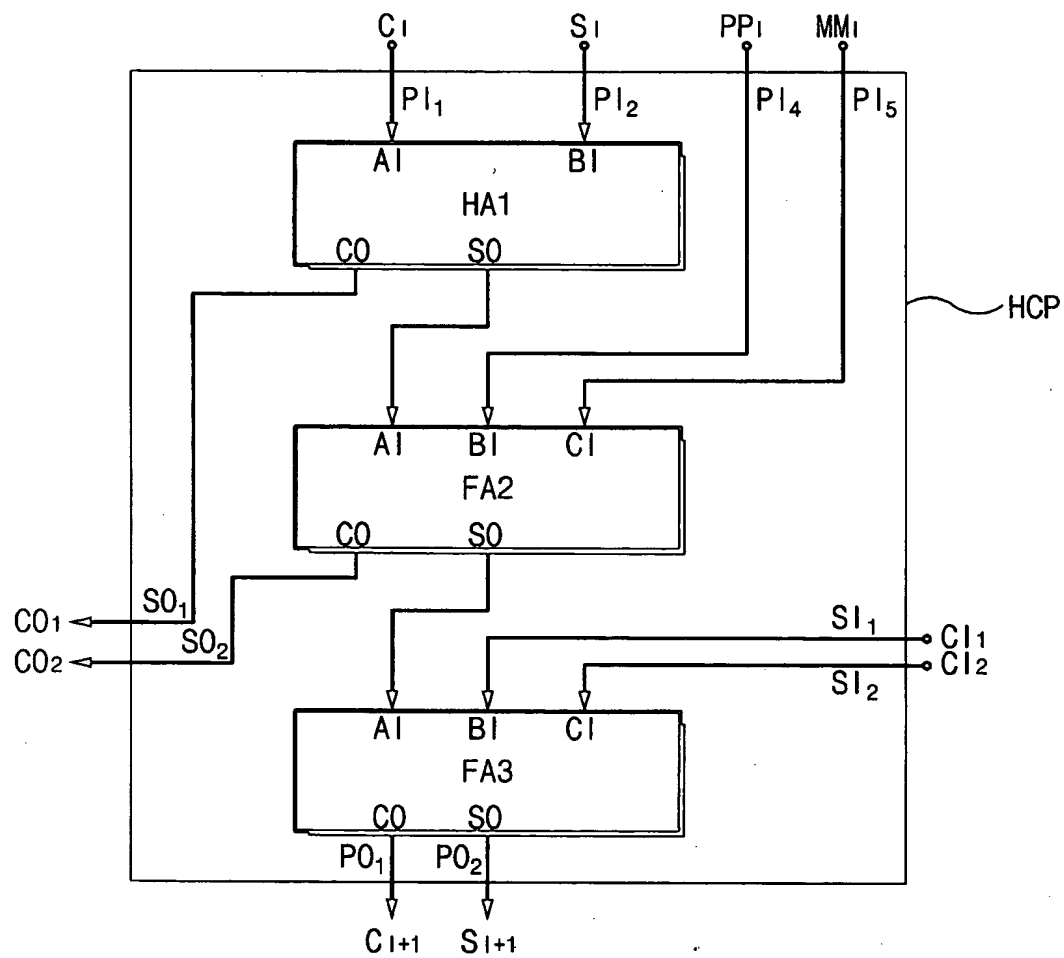


【도 2】

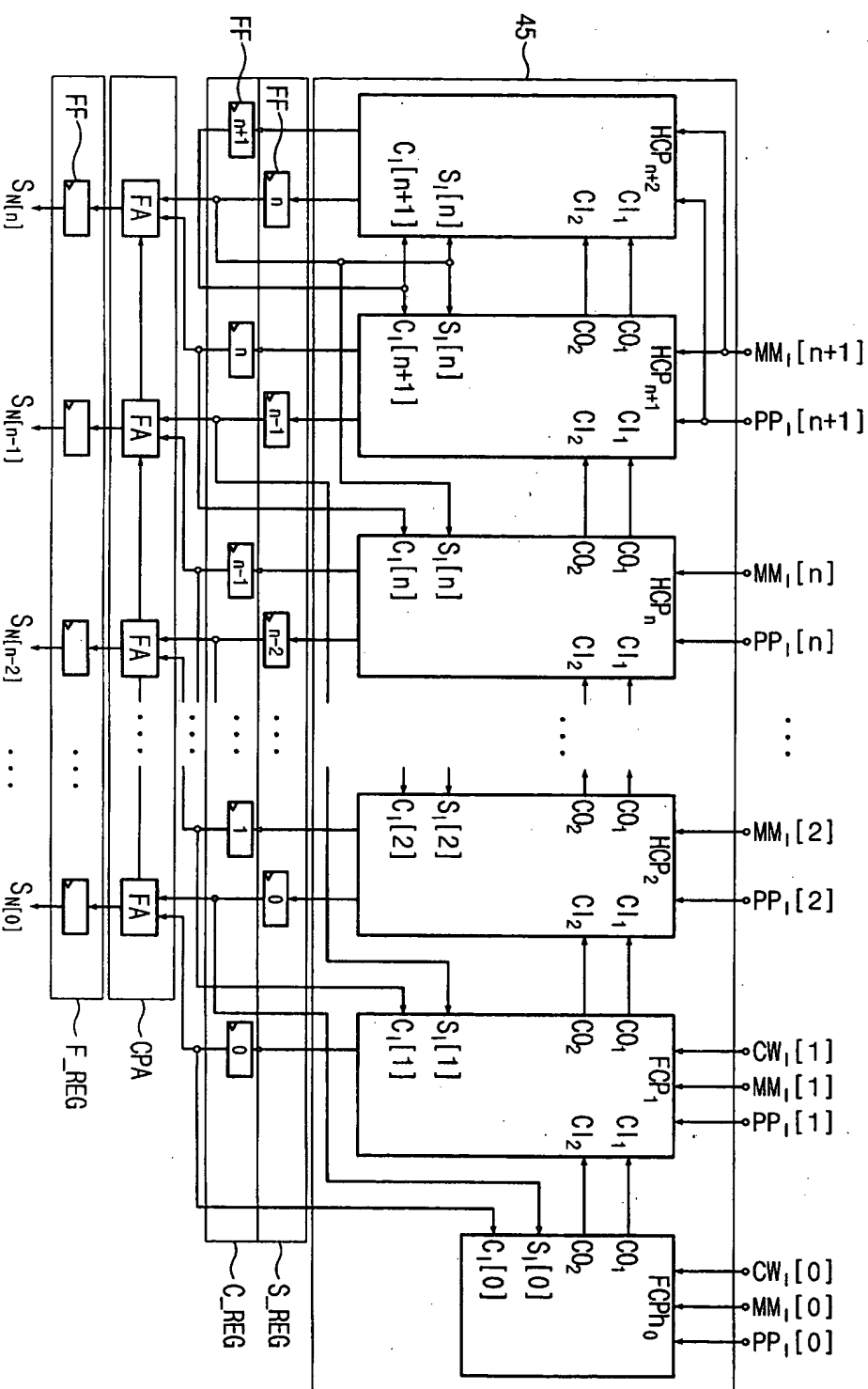




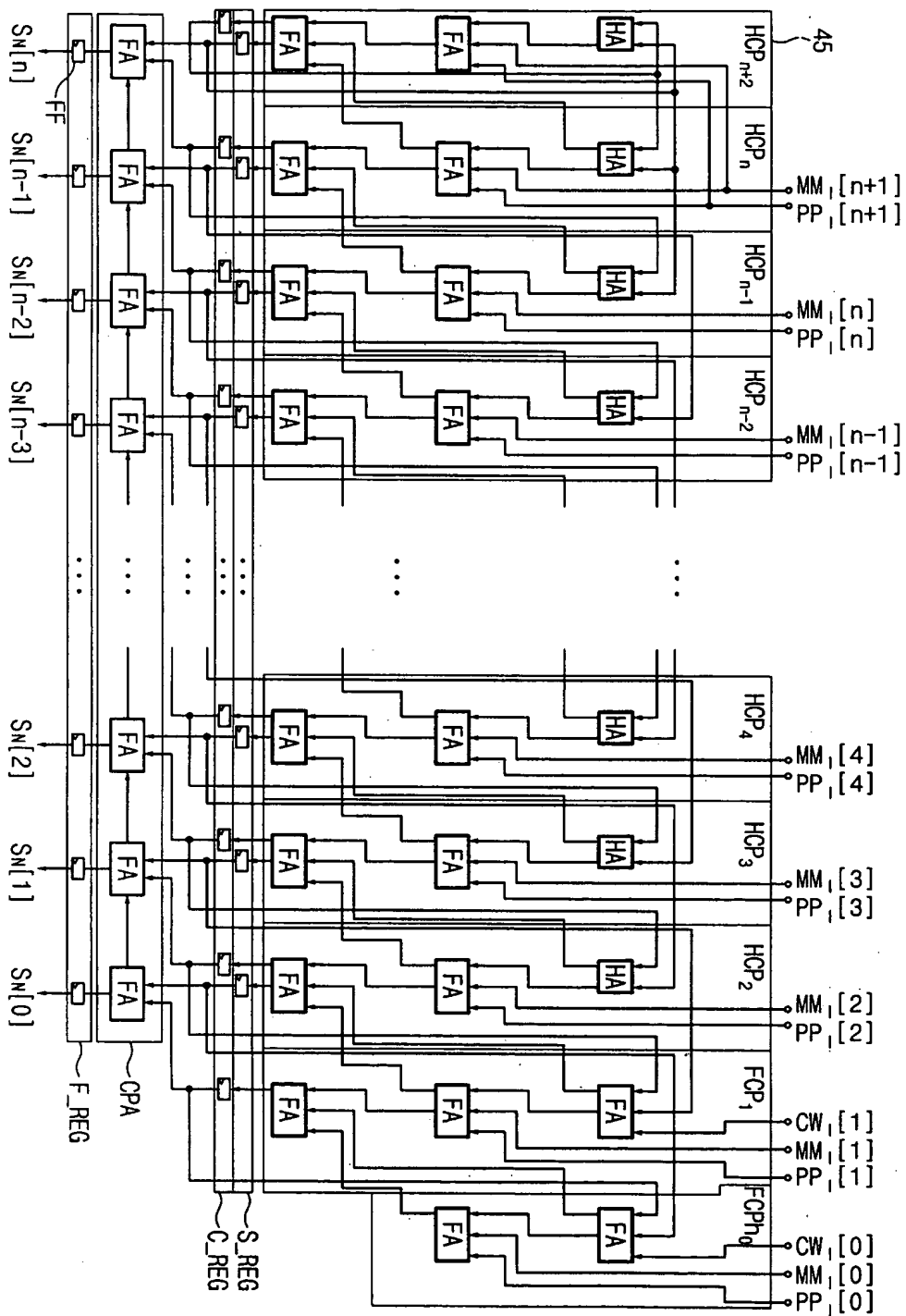
【도 3】



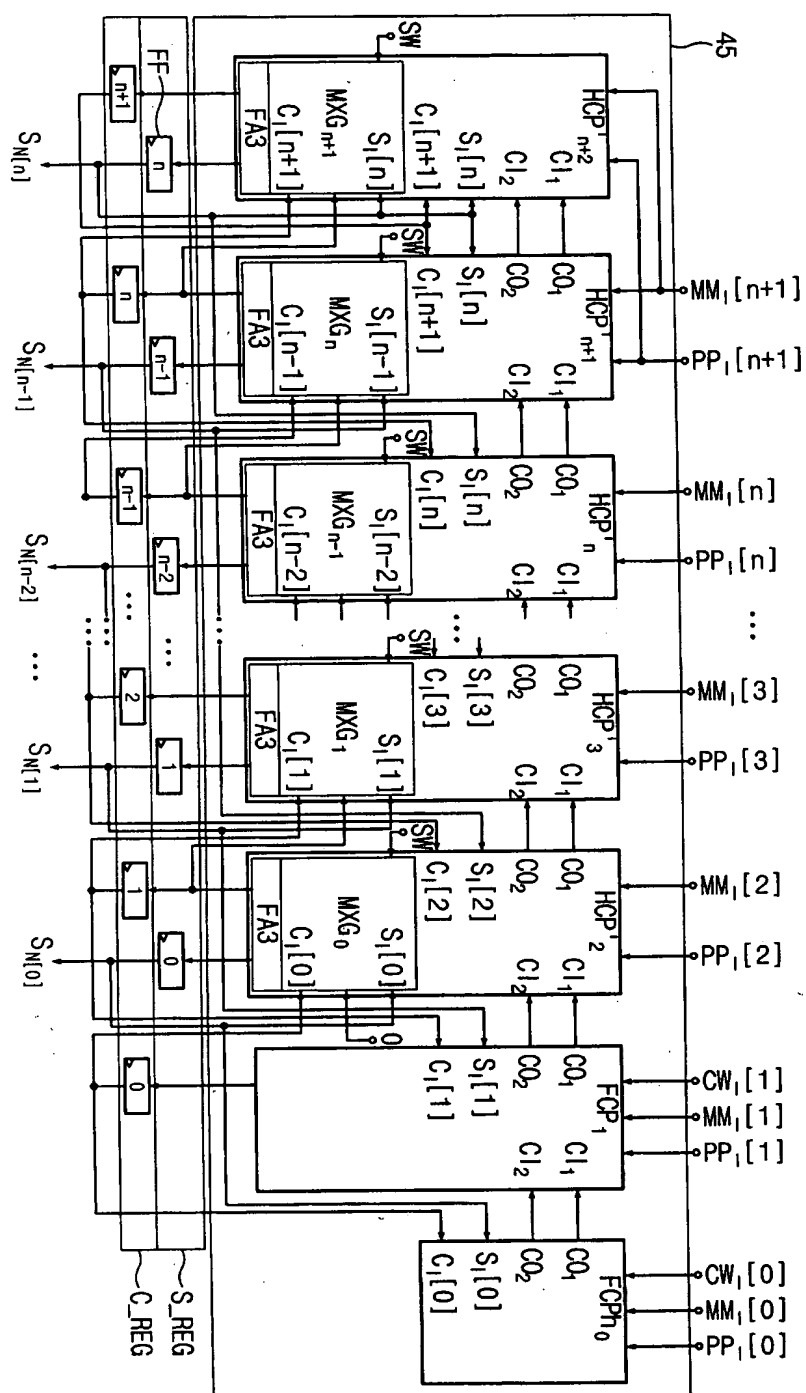
【도 4】



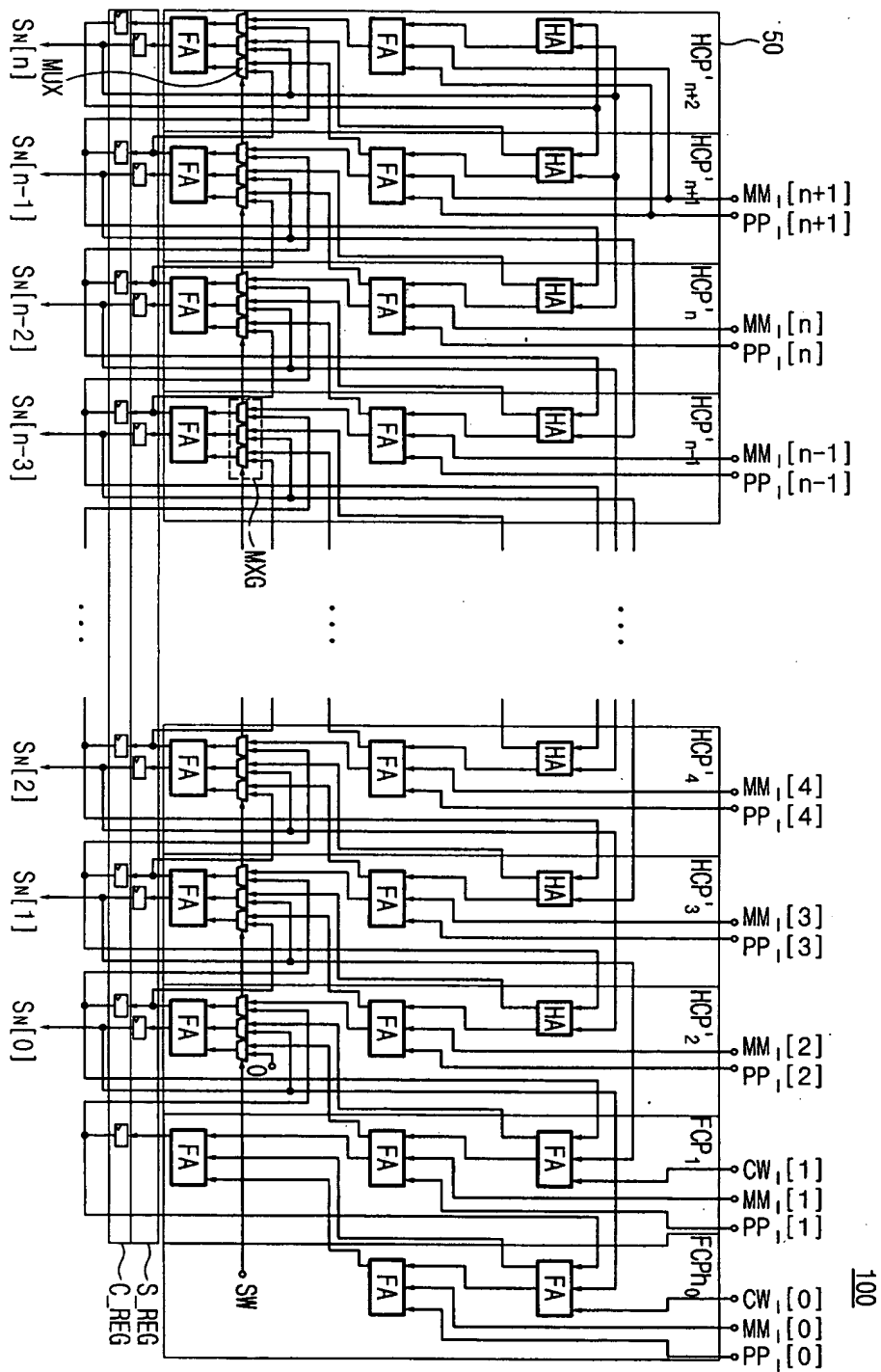
【도 5】



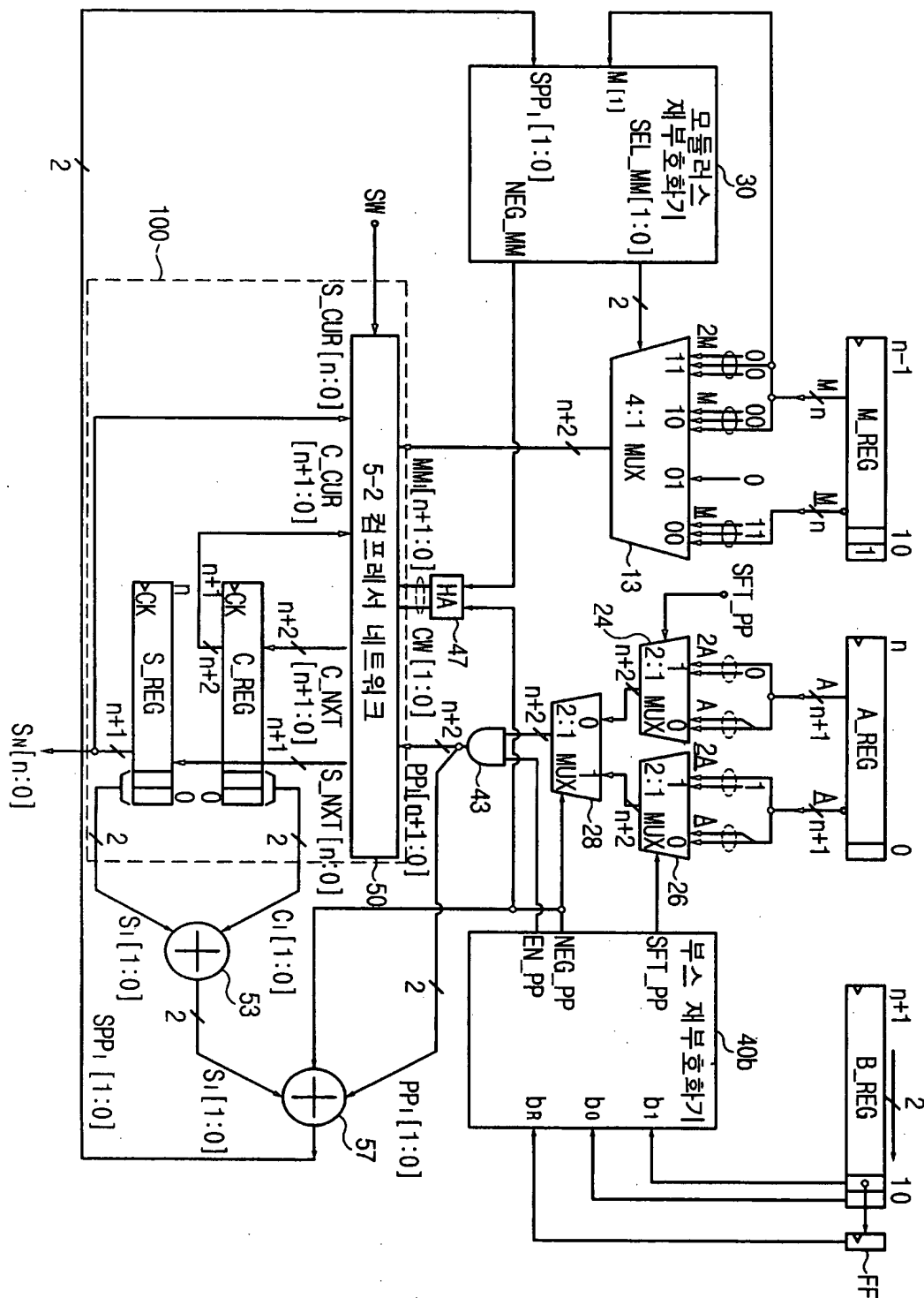
【도 6】



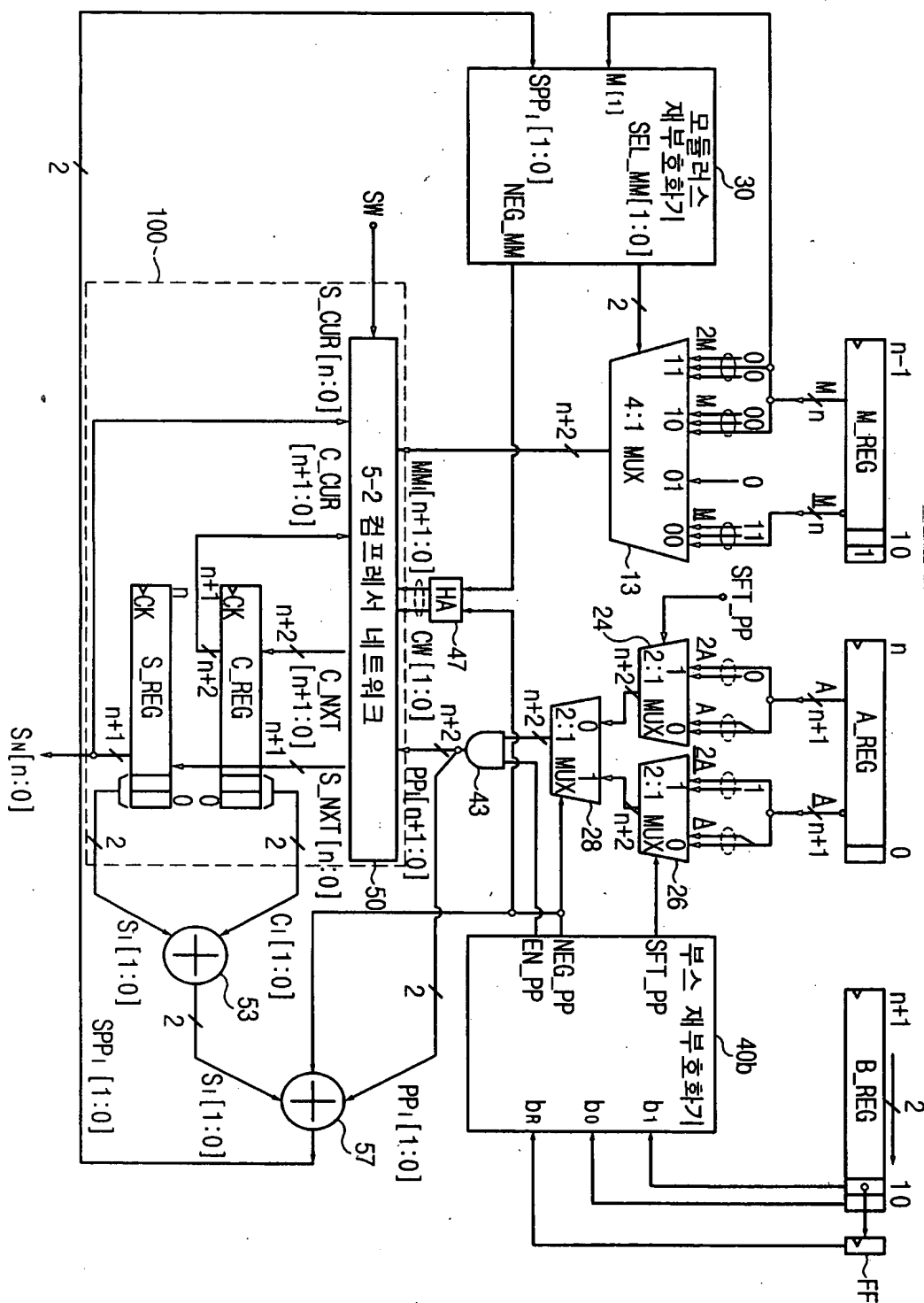
【도 7】



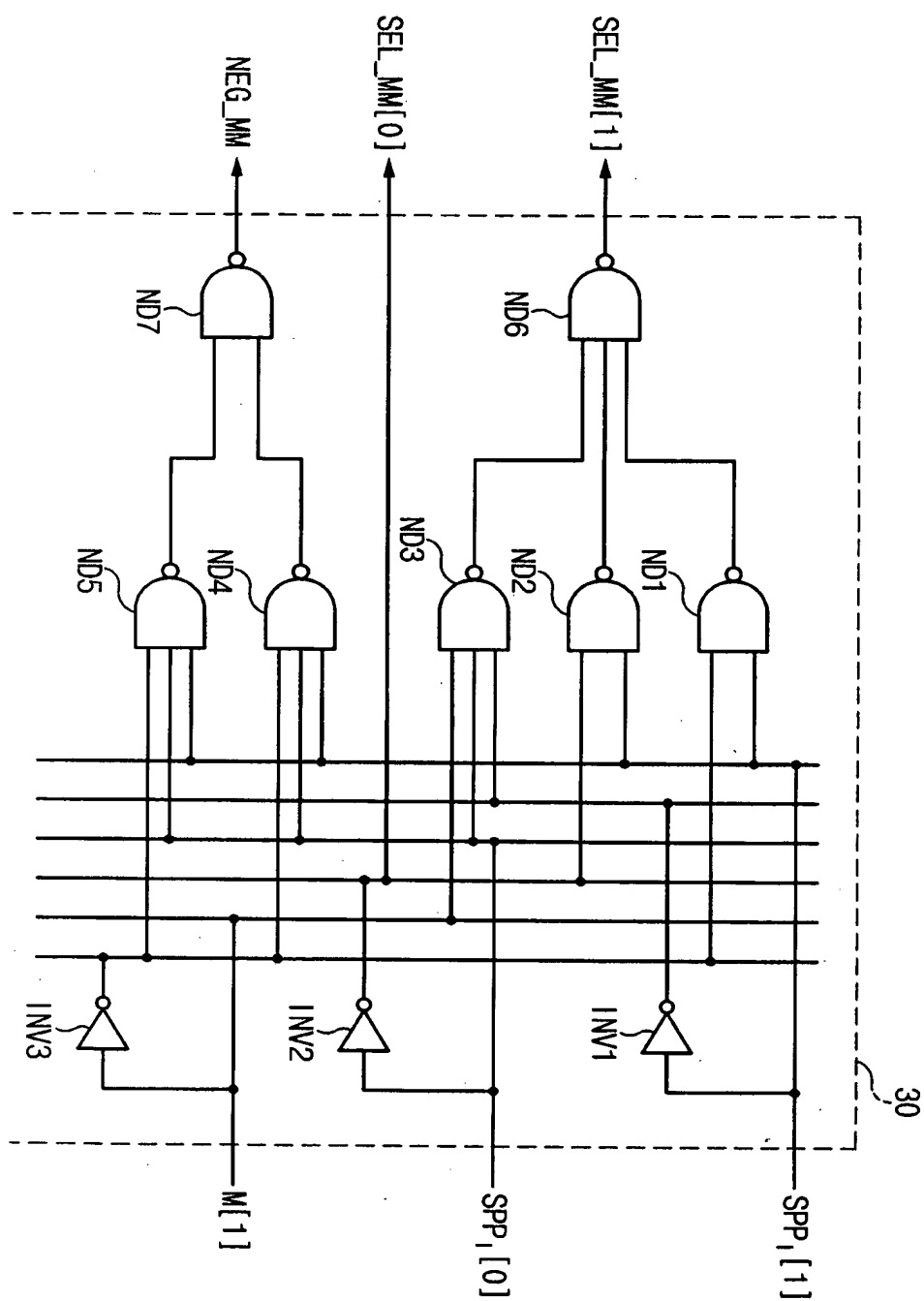
【도 8】



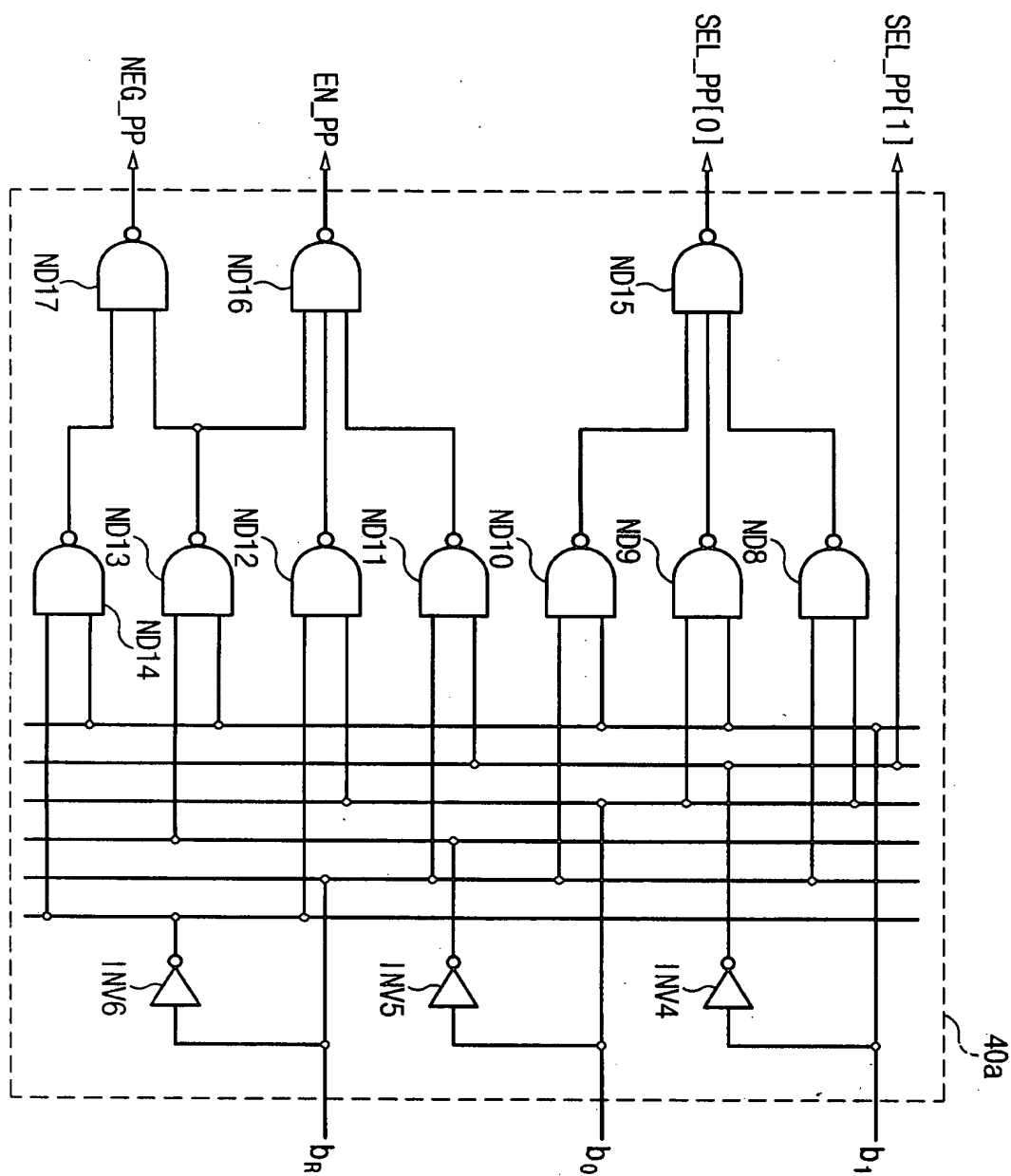
【도 9】



【도 10】



【도 11】



【도 12】

